République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Constantine 2 - Abdelhamid Mehri
Faculté des Nouvelles Technologies de l'Information et la Communication
Département de Technologies des Logiciels et des Systèmes d'Information
Laboratoire LIRE

N° d'ordre :
Série :



THÈSE

pour obtenir le grade de

DOCTEUR EN INFORMATIQUE

Option : Génie Logiciel

Spécification et Vérification Formelle des Systèmes Cloud

Présentée et soutenue par : Zakaria Benzadri
Dirigée par : Pr. Faiza Belala

Soutenue le : 04/10/2016

Jury :

Pr. Mohamed Benmohamed	Université Constantine 2	Président du jury
Pr. Faiza Belala	Université Constantine 2	Directrice de thèse
Pr. Larbi Sekhri	Université d'Oran	Examinateur
Pr. Allaoua Chaoui	Université Constantine 2	Examinateur
Dr. Hammadi Bennoui	Université de Biskra	Examinateur
Dr. Nabil Hameurlain	Université de Pau et des Pays de l'Adour	Invité

Si vous avez construit des châteaux dans les nuages, votre travail n'est pas vain ; c'est là qu'ils doivent être. À présent, donnez-leurs des fondations.

Henry David Thoreau
Philosophe américain, 1817-1862.

À l'Eternel "ALLAH" pour m'avoir donné le courage, la force et la patience d'achever ce modeste travail.

À mes très chers parents pour leur amour inépuisable, leur patience et leur sacrifice. Que Dieu leur donne santé et longue vie.

À mon frère "Chouaib" et mes sœurs "Sara et Fatima" pour la fraternité et l'affection.

À mes beaux frères "Charaf Eddine et Zine El Abidine" pour leur encouragement.

À mon oncle "Mustapha" et son épouse "Nadjeh" pour leur soutien inconditionnel.

À ma chère famille pour les liens de solidarité qui nous réunissent.

À mes très chers amis "Nadir", "Taher" et "Nassim" pour leur gentillesse, et leur infinie disponibilité toutes les fois que j'ai eu besoin d'eux.

À tous ceux qui me sont chers.

Je dédie ce modeste travail.

Zakaria BENZADRI.

Remerciements

Je tiens à exprimer ma gratitude envers ma directrice de thèse, Pr. Faïza Belala, pour avoir accepté de diriger mes recherches. Merci Madame pour votre disponibilité permanente, confiance, rigueur, motivation, expérience, et pour m'avoir mis dans les conditions idéales pour réaliser ce projet, tant dans un contexte professionnel que scientifique riche.

J'exprime également ma gratitude envers Dr. Chafia Bouanaka pour avoir participé à l'avancement de mes recherches. Merci Madame pour votre grande disponibilité, patience, écoute, encouragement, compétence, et pour votre assistance plus que significative tout au long de ce projet.

Je remercie chaleureusement Dr. Nabil Hameurlain pour m'avoir accueilli dans son équipe MOVIES. Merci Monsieur pour votre grande bienveillance et vos précieux conseils.

Je remercie très sincèrement Pr. Mohamed Benmohamed, Pr. Larbi Sekhri, Pr. Allaoua Chaoui et Dr. Hammadi Bennoui pour m'avoir fait l'honneur de participer à mon jury de thèse. Qu'ils trouvent ici l'expression de mon profond respect.

J'adresse toute ma reconnaissance à tous mes enseignants pour avoir veillé à me donner du savoir tout au long de mon parcours scolaire et universitaire. Merci tout particulièrement aux enseignants de la faculté NTIC, et à mes enseignants remarquables Dr. Aicha Choutri, Dr. Nadira Benlahrache et Dr. Nadia Zeghib pour leurs conseils et leurs encouragements.

Je remercie très affectueusement les membres du laboratoire LIRE, particulièrement, son directeur Pr. Mahmoud Boufaida pour l'accueil et les sages conseils. Merci également à mes amis docteurs et doctorants du laboratoire LIRE pour tous les agréables moments que nous avons partagés.

Résumé

Le Cloud Computing consiste à proposer les ressources informatiques sous forme de services à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui. Très vite, ce nouveau paradigme attire beaucoup d'attention et devient aujourd'hui le modèle le plus utilisé pour héberger et exécuter les services informatiques. Néanmoins, et en raison de leur hétérogénéité, le degré de complexité des systèmes Cloud augmente de façon inexorable, et engendre une définition indéterminée des éléments impliqués dans une architecture Cloud. L'objectif principal de cette thèse est de fournir un niveau d'abstraction aux descriptions architecturales d'un système Cloud en mettant en place un modèle générique et modulaire qui permet de modéliser les différents éléments d'une architecture cloud, vu selon toutes ses facettes. Les bigraphes forment le moyen, le plus approprié, pour une telle description du système cloud ainsi que son comportement. Nous les adoptons dans le contexte de ce travail, pour la première fois dans la littérature, afin de proposer une approche formelle originale, pour la spécification et la vérification des systèmes Cloud. Nous définissons un modèle théorique (CAB –Cloud Architecture Bigraph) prenant en charge toutes les spécificités d'une architecture Cloud, et permettant la modélisation de ces reconfigurations dynamiques. Une conséquence directe et importante qui s'est découlée de cette formalisation consiste en l'association d'une sémantique formelle (CScB –Cloud Services composition Bigraph) à la composition dynamique des services Cloud. Par ailleurs, nous concrétisons tous les résultats théoriques obtenus, via l'implémentation d'un outil générique (RCTool4Bigraphs) dédié à l'édition, la réécriture et la vérification basée "Model-Checking" des bigraphes. Nous avons pour cela exécuté et analysé formellement les deux modèles proposés : CAB et CScB.

Mots-clés : Cloud Computing ; Architecture Cloud ; Composition des Services Cloud ; Bigraphes ; CAB ; CScB ; RCTool4Bigraphs.

Abstract

Cloud computing is actually attracting more attention, as a promising model for delivering Information and Communication Technology (ICT) services via the generalization of service reuse to all computer resources. It involves dynamic and on demand provisioning of shared computing resources. Albeit, this new paradigm has evolved considerably in the last few years and its application areas are becoming very numerous, the degree of complexity in cloud systems is increasing inexorably due to their heterogeneity. An important and challenging issue in this area, is how to associate a clear semantic to cloud architecture concepts. Relying heavily on mathematical models, Bigraphs theory represents an effective approach to provide a clear and precise definition of cloud computing architecture. The present work is concerned with the use of Bigraphs theory for modelling and analysing cloud systems. We define a theoretical model (CAB –Cloud Architecture Bigraph) specifying the static structure of cloud architecture and allowing to represent its dynamic evolution. A nice consequence of this formalisation is the association of a formal semantic (CScB –Cloud Services composition Bigraph) to cloud services composition. Furthermore, we implement a generic tool (RCTool4Bigraphs) dedicated to editing, rewriting and model-checking bigraphs. To validate the obtained theoretical results, we use the proposed tool to formally execute and analyse the two models: CAB and CScB.

Keywords: Cloud Computing; Cloud Architecture; Cloud Service Composition; Bigraphs; CAB; CScB; RCTool4Bigraphs.

ملخص

الحوسبة السحابية تقوم على مبدأ توفير موارد الحوسبة كخدمة عند الطلب، يمكن الولوج إليها من أي مكان وفي أي وقت ومن قبل أي شخص. سرعان، ما اجتذب هذا النموذج الجديد اهتماما كبيرا، وأصبح الآن النموذج الأكثر استخداما على نطاق واسع لاستضافة وتشغيل تكنولوجيا المعلومات. ومع ذلك، لا تزال هناك العديد من التحديات التي تبطء وثيرة اعتماده ونموه. السؤال المهم والصعب في هذا السياق هو كيفية وضع أطر واضحة للمفاهيم الأساسية المرتبطة بالحوسبة السحابية. استنادا إلى نظرية الرسوم البيانية الثنائية (bigraphes) ، تقدم هاته الأطروحة إطار رسمي (CAB) يوفر التعريفات الرياضية اللازمة للعناصر الرئيسية التي تشارك في الحوسبة السحابية ، و التي تحدد بدورها هيكلها الثابت وتطورها الديناميكي. كنتيجة مباشرة وهامة لهذا الطابع الرسمي تقدم هاته الأطروحة أيضا إطار رسمي (CScB) لتكوين الخدمات السحابية. وعلاوة على ذلك، فإننا نطبق كل النتائج النظرية التي تم الحصول عليها من خلال اقتراح أداة (RCTool4Bigraphs) عامة مخصصة لتحرير، وإعادة كتابة والتحقق على أساس "Model-Checking" من الرسوم البيانية الثنائية. و اسنادا على هذا، قمنا بتنفيذ وتحليل النموذجين: CAB وCScB.

كلمات البحث: الحوسبة السحابية ; تكوين الخدمات السحابية ; الرسوم البيانية الثنائية.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La pensée informatique, telle qu'elle s'est introduite par son grand père "al-Khwarizmi", est l'approche scientifique et pratique pour désigner les techniques de calcul écrit et leurs applications [Van Der Waerden, 2013]. Depuis cela, cette pensée a subi de nombreuses évolutions et devient aujourd'hui de plus en plus puissante, de plus en plus présente, avec des usages diverses. L'informatique en nuage "Cloud Computing" constitue l'évolution la plus importante et la plus récente de la pensée informatique. Basée sur les notions de l'informatique en grille "Grid Computing", la principale originalité de l'informatique en nuage est de renforcer l'idée de l'informatique utilitaire "Utility Computing". Le "Cloud Computing" consiste à proposer les ressources informatiques sous forme de services à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui. Très vite, ce nouveau paradigme informatique attire beaucoup d'attention de l'industrie et du milieu académique.

Contexte et Problématique

L'année 1961 a vu l'apparition de la notion d' "Utility Computing" –introduite par John McCarthy[Foster et al., 2008]. Cette vision futuriste consiste à vendre la puissance de calcul et même les applications spécifiques comme un service public. A l'époque, l'idée novatrice de la consommation des services informatiques n'a pas vu le jour, car, les technologies matérielles et logicielles n'étaient tout simplement pas prêtes. De nos jours, et grâce aux avancées des technologies de l'information et de la communication, cette idée a incarné à nouveau, elle est apparue sous une autre forme. Le Cloud Computing met en œuvre l'idée d' "Utility Computing" en offrant une allocation dynamique des ressources informatiques selon trois modèles de prestations de service : l'infrastructure en tant que service (IaaS), la plateforme en tant que service (PaaS), et l'application en tant que service (SaaS). D'un point de vue déploiement, le Cloud Computing propose aussi quatre modèles : l'infrastructure est exploitée pour le compte d'une entreprise –Cloud privé ; l'infrastructure est partagée entre plusieurs entreprises –Cloud communautaire ; l'infrastructure est mise à la disposition du grand public –Cloud public ; et enfin, l'infrastructure est une combinaison de deux ou plusieurs

des modèles de déploiements précédents –Cloud hybride.

L'émergence du Cloud Computing représente la prochaine évolution dans les architectures distribuées [Marks and Lozano, 2010], il devient aujourd'hui le modèle informatique le plus utilisé pour héberger et exécuter les services informatiques. Cette popularité, de plus en plus croissante, est due à ses nombreux avantages et caractéristiques tels que, réduire le coût total de possession des systèmes informatiques (offre modulable en coût –en fonction de la demande réelle), améliorer l'accessibilité et la flexibilité (notamment pour les organisations multi-sites), réduire les efforts de gestion informatique (facilité et rapidité de déploiement), élasticité, évolutivité et mobilité, etc. Néanmoins, et en raison de leur hétérogénéité, le degré de complexité des systèmes Cloud augmente de façon inexorable [Bessis et al., 2012], et engendre une définition indéterminée des éléments impliqués dans une architecture Cloud. Plusieurs défis sont apparus freinant la croissance et l'adoption du Cloud Computing [Armbrust et al., 2009].

Nous considérons dans ce travail les trois défis suivants :

D1 : Bugs dans les grands systèmes distribués. Il est difficile d'anticiper et éviter les bugs au cours du développement des systèmes Cloud. C'est encore aussi difficile de les trouver et les corriger dans un système en cours d'exécution.

D2 : Disponibilité d'un service. La disponibilité est un enjeu important des systèmes Cloud. Elle est mesurée par rapport à la perception des utilisateurs d'un service Cloud.

D3 : Mise à l'échelle rapide. La mise à l'échelle rapide des systèmes Cloud constitue un autre défi majeur en réponse au nombre de demandes des utilisateurs dans le Cloud.

Notons que ces défis considérés se chevauchent et peuvent engendrer d'autres problèmes actuellement connus dans le Cloud. Les "bugs" peuvent provoquer une dégradation significative des performances d'un système Cloud, et rendre ainsi ses services non-disponibles. Pour surmonter ces défis (*D1*, *D2* et *D3*), la modélisation et l'analyse des systèmes Cloud représente une meilleure solution. Par ailleurs, et à défaut de pouvoir réduire la complexité des systèmes Cloud en général, il a fallu la maîtriser en donnant une illusion de simplicité. Cette illusion est possible grâce à l'abstraction, qui permet de réduire la quantité d'informations. D'une part en mettant en avant les caractéristiques essentielles du système, selon toutes ses facettes. D'autre part en ignorant ses caractéristiques secondaires. Les méthodes formelles forment le moyen, le plus approprié, pour une telle description du système cloud ainsi que son comportement. De plus, elles semblent les mieux

adaptées pour la vérification des systèmes Cloud, en fournissant le plus haut niveau d'abstraction et d'assurance de l'évaluation "EAL7" –selon le standard international pour la sécurité des systèmes d'information [Standard, 2009].

En réalité, les études menées sur la modélisation et l'analyse des systèmes Cloud montrent que la plupart de ces approches sont ad-hoc, complexes, difficiles à adopter et à réutiliser. Aux premiers stades de la recherche sur le Cloud Computing, les différentes approches proposées étaient principalement des approches à intérêt industriel (ou technique). Ces approches offrent des solutions spécifiques pour traiter un problème particulier par rapport à une technologie imposée par un fournisseur Cloud, ou bien un besoin signalé par un utilisateur. Néanmoins, au cours de ces dernières années, de nouvelles approches à caractère académique sont développées. Ces approches proposent des solutions abstraites mais non-génériques ; les formalismes utilisés ne sont pas dédiés à la modélisation des systèmes Cloud et offrent une seule "vue " du système –qui n'est pas suffisante.

Objectifs et Contributions

Cette thèse s'inscrit dans le cadre de la spécification et la vérification des systèmes Cloud, sa contribution principale est d'adopter, à un niveau plus abstrait, une meilleure séparation entre les différentes couches d'une architecture Cloud : couche utilisateurs, couche services et couche virtualisation d'une part, et d'exploiter le formalisme des bigraphes d'autre part, le plus approprié pour faciliter la conception, la réutilisation et le raisonnement sur l'évolution du système Cloud. En effet, la théorie des bigraphes [Milner, 2009], constituant un formalisme unificateur de plusieurs modèles de concurrence, permet de modéliser deux vues importantes des systèmes distribués en général : (1) *"Mobile Locality"* –distribution spatiale des entités physiques ou logicielles, et (2) *"Mobile Connectivity"* –interaction entre l'ensemble de ces entités. Cette théorie offre de plus la possibilité de décrire l'évolution dynamique d'un système à l'aide d'un ensemble de règles de réaction, constituant ainsi les "Systèmes Réactifs Bigraphiques" (BRS).

Notre objectif est de proposer des modèles qui soient suffisamment expressifs (supportant les éléments de base d'architecture Cloud) tout en restant raisonnablement analysables (permettant d'assurer les propriétés relatives aux systèmes Cloud). Des éléments de réponses aux challenges (*D1*, *D2* et *D3*) cités précédemment sont alors apportés :

1. Un modèle théorique supportant les éléments fondamentaux d'une architecture Cloud. Ceci permet d'abstraire la réalité des systèmes Cloud et de maitriser leur complexité (en réponse à *D1*).

2. Des relations générales décrivant l'évolution dynamique d'une architecture Cloud et permettent de mieux comprendre et prévoir le comportement d'un système Cloud (en réponse à *D3*).
3. Un raisonnement sur les comportements possibles d'un système Cloud en vue de prouver certaines propriétés –telle que la disponibilité (en réponse à *D2*).

Nous adoptons dans le contexte de ce travail, et pour la première fois [Benzadri et al., 2013a], la théorie des bigraphes afin de définir :

— Un modèle bigraphique (CAB –*"Cloud Architecure Bigraph"*) prenant en charge toutes les spécificités d'une architecture Cloud, composé de trois modèles bigraphiques différents : (i) le CUB (*"Cloud Users Bigraph"*) supportant l'ensemble des concepts de la couche utilisateurs ; (ii) le CSB (*"Cloud Services Bigraph"*) décrivant les différents éléments de la couche services ; et (iii) le CVB (*"Cloud Virtualization Bigraph"*) formalisant les entités de base de la couche virtualisation.

— Un ensemble de méta-règles de réaction pour la modélisation des reconfigurations dynamiques au niveau des trois couches identifiées précédemment.

— Un méta-modèle décrivant les différents concepts proposés autour de la composition des services Cloud et projeter ainsi ces définitions dans le cadre de la théorie des bigraphes (CScB –*"Cloud Services composition Bigraph"*). Une sémantique formelle est alors associée à la composition dynamique des services Cloud selon deux vues distinctes mais complémentaires : *"Composition Verticale"* et *"Composition Horizontale"*.

— Une méthodologie de spécification et vérification des systèmes cloud, concrétisée via le développement d'un outil générique (RCTool*4*Bigraphs) dédié à l'édition, la réécriture et la vérification basée "Model-Checking" des bigraphes.

Organisation du manuscrit

Le manuscrit est principalement structuré en deux grandes parties contenant six chapitres, en plus d'une introduction et conclusion générales ;

— **La première partie** (*État de l'art*), présente une synthèse récapitulative des différents concepts liés au cadre de ce travail et explore les approches jugées intéressantes de la modélisation des systèmes Cloud. Elle comprend trois chapitres :

  — **Le chapitre 1** (*Principe et Concepts du Cloud Computing*), est consacré à la présentation des aspects importants du Cloud Computing. En s'aidant des définitions les plus adoptées du Cloud, une définition

intégrée permettant une poursuite aisée de ce travail est dégagée et donnée dans ce chapitre.
  - **Le chapitre 2** (*Approches de Modélisation des Systèmes Cloud*), décrit de façon détaillée une classification des approches de modélisation existantes des systèmes Cloud afin de situer la contribution principale de ce travail. Il distingue entre deux grandes classes d'approches, celles à intérêt industriel et d'autres à caractère académique.
  - **Le chapitre 3** (*Théorie des Bigraphes*), consiste en une étude approfondie portant sur la théorie des bigraphes. Il décrit les constituants d'un bigraphe, ses principales opérations de composition, ses règles de réaction et les outils pratiques autour de ce formalisme.
- **La deuxième partie** (*Contribution*), représente la partie principale de ce travail où la sémantique des modèles proposés est donnée et validée à travers des études de cas ; l'organisation de cette partie est comme suit :
  - **Le chapitre 4** (*Formalisation d'Architecture Cloud*), comprend la définition du modèle bigraphique pour la spécification d'architecture Cloud –baptisé (*CAB*), ses trois bigraphes constituants (*CUB*, *CSB* et *CVB*), et l'ensemble de ses règles de réaction. Ce chapitre illustre également la validation de cette démarche à travers une étude de cas du système *"Cloud-Healthcare System"*.
  - **Le chapitre 5** (*Formalisation de la Composition des Services Cloud*), présente la définition du modèle bigraphique traitant de manière formelle la composition dynamique des services Cloud (le bigraphe *CScB* et ses règles de réaction). Cette démarche est illustrée en servant d'une étude de cas d'un système réel : *"Cloud Emergency Response System"*.
  - **Le chapitre 6** (*"RCTool4Bigraphs" : Outil pour la Réécriture et la Vérification des Bigraphes*), présente les phases d'implémentation de l'outil générique proposé ainsi que ses principales fonctionnalités. Il décrit également les résultats d'exécution et de vérification formelle des modèles bigraphiques (*CAB* et *CScB*), en utilisant les deux études de cas présentées (*"Cloud-Healthcare System"* et *"Cloud Emergency Response System"*).
- **La conclusion générale** (*Synthèse*), récapitule les contributions réalisées et propose des perspectives liées à la poursuite de ce travail.

Diffusion scientifique

Les différents travaux présentés dans ce manuscrit ont fait l'objet de diverses publications listées ci-dessous.

Article de Journal International :

— BENZADRI, Z., BOUANAKA, C. et BELALA, F. **"Big-CAF : a Bigraphical-generic Cloud Architecture Framework"**. Accepted to appear in *International Journal of Grid and Utility Computing*. 2016. Inderscience.

Chapitre de Livre :

— BENZADRI, Z., BOUANAKA, C. et BELALA, F. **"A Formal Framework for Cloud Systems"**. *Delivery and Adoption of Cloud Computing Services in Contemporary Organizations*, 245. 2015. IGI Global.

Conférences Internationales :

— BENZADRI, Z., BELALA, F. et BOUANAKA, C. **"Towards a Formal Model for Cloud Computing"**. *International Conference on Service Oriented Computing –ICSOC Workshops*, Berlin, Germany, December 2-5, 2013. Springer International Publishing : 381-393.

— BENZADRI, Z., BOUANAKA, C. et BELALA, F. **"On Specifying and Verifying Cloud Systems"**. *Verification and Evaluation of Computer and Communication Systems –VECoS*, Florence, Italy, November 21-22, 2013. eWiC series (ISSN 1477-9358) of the British Computer Society.

— BENZADRI, Z., BOUANAKA, C. et BELALA, F. **"Verifying Cloud Systems using a Bigraphical Maude-based Model Checker"**. *International Conference on Cloud Computing and Services Science –CLOSER Workshops*, Barcelona, Spain, April 03-05, 2014. SCITEPRESS Digital Library.

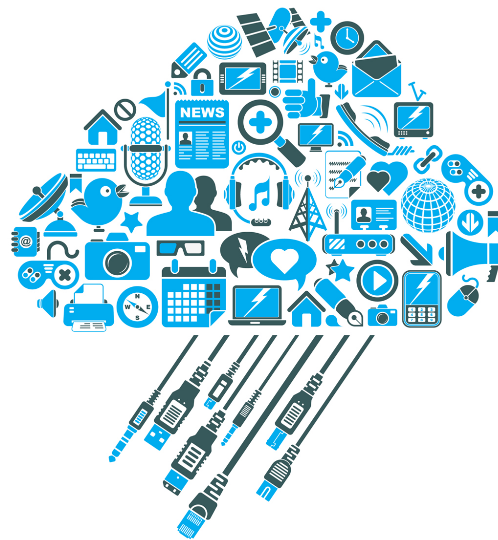
— BENZADRI, Z., BOUANAKA, C. et BELALA, F. **"BiCloud-2M : A Combined Bigraph Maude-based Tool for Cloud Specification and Analysis"**. *International Conference on Advanced Aspects of Software Engineering –ICAASE*, Constantine, Algeria, November 02-04, 2014. CEUR.

Première partie

État de l'art

Chapitre 1

Principe et Concepts du Cloud Computing



”The interesting thing about cloud computing is that we’ve redefined cloud computing to include everything that we already do. I can’t think of anything that isn’t cloud computing with all of these announcements”. –Larry Ellison, chairman, Oracle.

Table des matières

1.1 Introduction

Le Cloud Computing est un paradigme émergeant qui consiste à externaliser les ressources informatiques vers des centres de données performants, accessibles sous forme de services cloud et disponibles sur Internet. Ce nouveau paradigme chevauche avec de nombreuses technologies existantes [Foster et al., 2008]. Inspiré du modèle ASP ("*Application Service Provider*"), l'originalité du Cloud Computing repose sur la notion d' "*Utility Computing*". En effet, l'idée principale qui anime le développement du Cloud Computing a émergé de retour dans l'année 1961, lorsque John McCarthy, connu comme l'un des pionniers de l'intelligence artificielle (dont il proposa le nom en 1955 [McCarthy et al., 2006]), suggéra que la notion d' "*Utility Computing*" pouvait construire un bel avenir dans lequel la puissance de calcul et la capacité de stockage des informations pouvaient être vendues comme un service public (tels que l'électricité ou l'eau). Cette idée, n'a pas vu le jour à l'époque car les technologies matérielles, logicielles et réseaux n'étaient tout simplement pas prêtes. En outre, vers la fin des années 1990, le succès grandissant du Web a permis l'émergence du modèle ASP. Ce modèle avait pour ambition de déporter les applications métiers chez des hébergeurs spécialisés, libérant ainsi les entreprises des contraintes de support et de maintenance des applications. Il promettait également aux éditeurs des logiciels hébergés des revenus récurrents obtenus sous forme d'abonnement. Encore, l'idée était certes intéressante, mais à l'aube du Web 1.0, le manque de maturité des applications Web n'a pas permis aux utilisateurs d'avoir accès à des interfaces clientes riches. Par ailleurs, les fournisseurs ASP étaient contraints d'installer une application lourde sur le poste client qu'il fallait maintenir. Toutefois, depuis l'an 2000 et avec le Web 2.0, le Cloud Computing a pris tout son sens. Officiellement, l'expression "Cloud Computing" est né le 24 août 2006 suite à l'annonce de l'offre "Amazone Elastic Computing Cloud" (AEC2) [Jennings, 2010]. Cet offre consiste à proposer des ressources informatiques flexibles ; à la demande des utilisateurs. Aujourd'hui, le Cloud Computing est devenu un domaine très actif que ce soit dans le secteur académique ou bien dans l'industrie.

Dans ce premier chapitre, nous dressons un état de l'art sur le Cloud Computing. Nous rappelons dans un premier temps les différentes définitions et caractéristiques du Cloud Computing. Nous présentons également une description des éléments techniques du Cloud, tout en évoquant ses différents domaines d'usage. Nous terminons ce chapitre par une conclusion qui met en lumière les défis encore à relever du Cloud Computing.

1.2 Définitions

Depuis 2007, de nombreux auteurs ont tenté de définir le Cloud Computing selon différents points de vues. Cela a mené à un débat ouvert sur le Cloud Computing et sa définition [Antonopoulos and Gillam, 2010, Wu, 2011] ; ses détracteurs l'accusent de n'être qu'un un effet de mode "buzz-word" à des fins de marketing, alors que ses partisans voient en lui le futur des applications distribuées et le considèrent comme un paradigme informatique très émergeant. Parmi les définitions des partisans du cloud, nous citons trois définitions qui sont largement utilisées dans l'industrie et le domaine académique :

Définition 1.1 *"Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [Mell and Grance, 2011].*

Définition 1.2 *"Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" [Buyya et al., 2009].*

Définition 1.3 *"Cloud Computing is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on-demand to external customers over the Internet" [Foster et al., 2008].*

En guise de synthèse des différentes définitions présentées, nous introduisons une définition intégrée [Benzadri et al., 2016], qui sera considérée lors de la réalisation de ces travaux de thèse ;

Définition 1.4 *"Cloud Computing is a large-scale distributed computing paradigm in which a pool of interconnected and virtualized resources are dynamically provisioned to customers via three service delivery models (SaaS, PaaS and IaaS) and four deployement models (Public, Private, Community and Hybrid)" [Benzadri et al., 2016].*

Comparée à d'autres définitions, cette définition offre une vue plus spécifique et objective du cloud computing. Nous insistons sur la vue académique du cloud en tant que *paradigme informatique* repris de la définition 1.3. Ensuite, nous introduisons la notion de *virtualisation* et nous mentionnons le concept des utilisateurs

dans le cloud ; les deux cités dans les définitions 1.2 et 1.3, et absents dans la définition 1.1. Enfin, nous soulignons les trois modèles de prestation des services cloud (*"SaaS", "PaaS" et "IaaS"*) ainsi que leurs modèles de déploiement *("Public", "Private", "Community" et "Hybrid")* ; qui sont à leur tour décrits dans la définition 1.1 et égarés dans les définitions 1.2 et 1.3. Dans les sections suivantes nous décrivons chacun de ces concepts fondamentaux.

1.3 Caractéristiques

Le cloud computing possède cinq caractéristiques essentielles [Mell and Grance, 2011] :

Libre-service à la demande : Un utilisateur peut se procurer unilatéralement et automatiquement des ressources informatiques (de calcul et/ou de stockage) ; en fonction de ses besoins et sans interaction humaine avec le fournisseur de services.

Accès étendu au réseau : Les ressources informatiques sont disponibles sur le réseau et accessibles par le biais de mécanismes normalisés, permettant de les exploiter sur des plates-formes hétérogènes à clients lourds ou légers (des téléphones mobiles, des ordinateurs portables, etc.).

Mutualisation des ressources : Les ressources informatiques sont mises à la disposition des utilisateurs en adoptant un modèle de multi-location, avec assignation et réaffectation dynamiques des ressources physiques et virtuelles en fonction de la demande. L'utilisateur ne contrôle ni ne connait l'emplacement exact des ressources. Ces ressources comprennent, par exemple, le stockage, des unités de traitement, la mémoire, la bande passante du réseau et les machines virtuelles.

Élasticité/mise à l'échelle rapide : Des ressources peuvent être fournies rapidement, de manière élastique et, dans certains cas, automatiquement, afin d'assurer une mise à l'échelle rapide. Elles peuvent être ajoutées ou libérées rapidement pour augmenter ou réduire l'échelle du système. Pour l'utilisateur, les ressources disponibles semblent souvent être infinies et peuvent être achetées dans n'importe quelle quantité et à tout moment.

Service mesuré : Les systèmes cloud contrôlent et optimisent automatiquement l'utilisation des ressources, grâce à un outil de mesure approprié au type de service utilisé. Ce type de systèmes met en avant le concept de "consommation à l'utilisation".

1.4 Éléments Techniques

Parmi les éléments de base du cloud computing [Hausman et al., 2013], nous intéressons plus particulièrement au :

Service cloud. Les services cloud sont mis à la disposition des utilisateurs, sur demande, par Internet. Ils sont conçus pour fournir un accès évolutif aux ressources informatiques, et sont entièrement gérés par un fournisseur cloud. Ces services possèdent trois modèles de prestation ("Infrastructure as a Service" –IaaS, "Platform as a Service" –PaaS et "Software as a Service" –SaaS), et quatre modèles de déploiement (cloud public, cloud privé, cloud communautaire et cloud hybride).

Utilisateur cloud. Le cloud computing offre un grand confort aux utilisateurs en leurs assurant un environnement de travail accessible depuis n'importe où ; que ce soit pour des utilisateurs mobiles depuis leurs domicile (gare, web café, aéroport, etc.), ou bien des utilisateurs présents directement dans leurs organisations. Un utilisateur cloud peut être défini comme étant une personne se servant du cloud computing pour l'utilisation de services. Tout comme il existe de nombreux modèles de services cloud, il existe plusieurs types d'utilisateurs dans le cloud. Selon le service demandé, trois types d'utilisateurs cloud sont identifiés [Hausman et al., 2013] : des administrateurs cloud accédant aux IaaS, des développeurs cloud utilisant les services offerts par les fournisseurs de PaaS, et enfin, des utilisateurs finaux consommant les applications fournies par les fournisseurs de SaaS. La figure 1.1 aligne ces rôles en utilisant un modèle en pyramide.

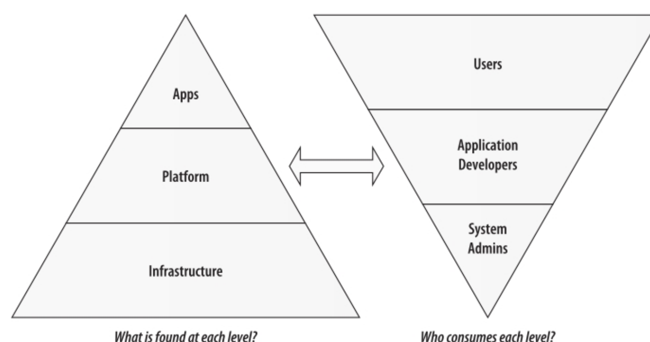


FIGURE 1.1 – Types d'Utilisateurs cloud [Hausman et al., 2013]

Fournisseur cloud. Un fournisseur cloud représente une entreprise basée sur la virtualisation des ressources informatiques, qui offre des solutions (infrastructures, plateformes ou applications) à d'autres entreprises et / ou individus, généralement via Internet.

Virtualisation cloud. La virtualisation est l'abstraction des ressources informatiques qui consiste à faire fonctionner de nombreux serveurs virtuels sur un seul serveur physique (voir la figure 1.2) [Ou, 2006]. Elle présente plusieurs intérêts non négligeables pour le cloud computing [Marks and Lozano, 2010], en particulier, la mutualisation des capacités d'un serveur cloud, la réduction de la consommation d'énergie des centres de données cloud, ainsi que la flexibilité et la rapidité d'hébergement des services cloud.

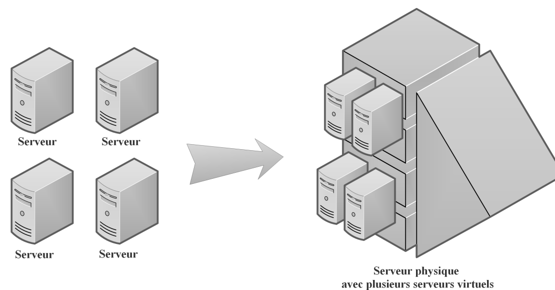


FIGURE 1.2 – Virtualisation des serveurs

Serveur cloud. Les serveurs cloud désignent des serveurs virtuels ou physiques, accessibles à distance, à partir d'un fournisseur cloud.

Cluster cloud. Un cluster représente un groupe de nœuds (serveurs ou services) qui travaillent ensemble pour maintenir la plus haute disponibilité du système cloud.

Centre de données cloud. Un centre de données cloud est un site physique qui regroupe des clusters de serveurs cloud, sur le quel il centralise les opérations constituant le système d'information d'une entreprise.

Fédération cloud. Les fournisseurs actuels du cloud possèdent plusieurs centres de données ("datacenters"), situés à différents endroits géographiques et accessibles uniquement via Internet. Cette dispersion permet de servir les besoins des utilisateurs dans le monde entier de façon optimale. Cependant, il s'est avéré que la coordination dynamique entre les différents centres de données, est une tâche d'extrême complexité. A cette fin, et en tirant parti du "Grid Computing", la notion de fédération dans le cloud vise à accroître la disponibilité des ressources cloud via la coordination non centralisée des services cloud distincts –en termes de fournisseurs et d'emplacement [Kurze et al., 2011]. Un nuage fédéré ("federated cloud") supporte deux scénarios de fédération [Kurze et al., 2011] : (1) *Migration*, permet la délocalisation des ressources, telles que les machines virtuelles et les services, d'un centre de données cloud à un autre ;

et (2) *Réplication*, permet l'utilisation simultanée de ressources cloud similaires, issues de différents centres de données. La fédération cloud peut être d'intérêt pour les fournisseurs, aussi bien que pour les utilisateurs. Les utilisateurs peuvent profiter des coûts réduits et de la meilleure performance, tandis que les fournisseurs peuvent augmenter la disponibilité de leurs services en cas de défaillance du système (par exemple, des attaques distribuées par déni de service). La figure 1.3 présente un exemple de fédération dans le Cloud.

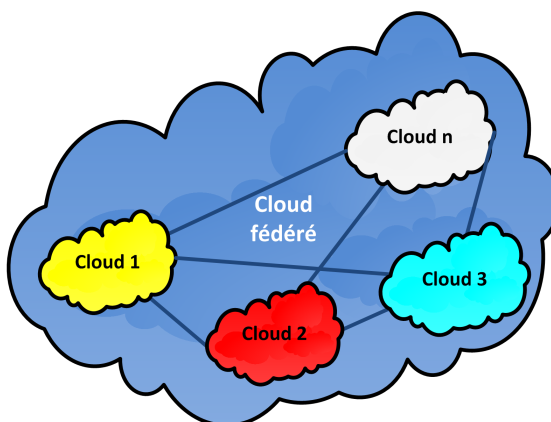


FIGURE 1.3 – Fédération Cloud

1.5 Services Cloud

La notion de service telle que définie par [SOAWorkGroup, 2013], est comme suit : "une représentation logique d'une activité, commerciale ou non, reproductible –formée d'un résultat spécifique et autonome, qui peut se composer d'autres services en cachant son implémentation interne". Un service cloud peut être défini comme étant un service fourni via Internet, avec la particularité d'être délivré en tant que commodité (comme la télévision, l'électricité ou le gaz), et selon trois modèles de prestation et quatre modèles de déploiement.

1.5.1 Modèles de Prestation

Le cloud computing comprend trois modèles de service qui collaborent pour assurer les demandes frontales : "Infrastructure as a Service" (IaaS), "Platform as a Service" (PaaS) et "Software as a Service" (SaaS).

"Infrastructure as a Service" (IaaS). Cette offre permet aux utilisateurs de se procurer de la capacité du traitement, de stockage, des réseaux et d'autres ressources informatiques essentielles. L'utilisateur est en mesure

de déployer et d'exécuter des logiciels tels que des systèmes d'exploitation et des applications. Il ne gère ni contrôle l'infrastructure cloud sous-jacente, mais il a le contrôle sur les systèmes d'exploitation, le stockage, les applications déployées et, éventuellement, certains composants réseau (par ex. des pare-feu).

"Platform as a Service" (PaaS). Cette offre permet aux utilisateurs de déployer leurs applications créées en interne ou achetées sur l'infrastructure cloud du fournisseur, dans la mesure où celles-ci sont basées sur des langages de programmation et des outils pris en charge par le fournisseur. L'utilisateur ne gère ni contrôle le réseau, les serveurs, les systèmes d'exploitation ou le stockage, mais il a le contrôle sur les applications déployées avec la possibilité de configurer l'environnement d'hébergement applicatif.

"Software as a Service" (SaaS). Cette offre permet aux utilisateurs d'accéder à des applications s'exécutant sur une infrastructure cloud à partir de divers appareils. L'utilisateur ne gère ni contrôle le réseau, les serveurs, les systèmes d'exploitation, le stockage, aussi bien que les fonctionnalités de chaque application, à l'exception d'une éventuelle configuration limitée et spécifique aux utilisateurs.

Ces modèles de services cloud constituent des "briques" bien définies, qui s'appuient les unes sur les autres ; la plateforme (PaaS) tire parti de l'infrastructure (IaaS), et l'application (SaaS) s'appuie sur la plateforme comme niveau de service [Marks and Lozano, 2010]. La figure 1.4 permet de décrire les différents niveaux de services cloud.

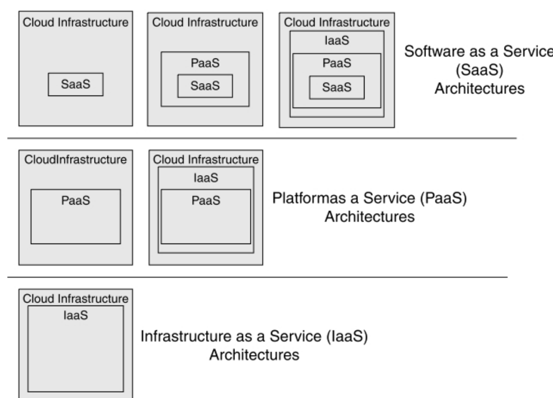


FIGURE 1.4 – Modèles de Services Cloud [Marks and Lozano, 2010]

D'autres déclinaisons des modèles de services cloud ont été utilisées par les pro-

fessionnels. Parmi ces déclinaisons : *"Data as a Service"* qui fournit des données de qualité à un endroit précis ; *"Business Process as a service"* qui consiste à externaliser une procédure d'entreprise suffisamment industrialisée ; *"UI as a Service"* qui fournit une interface complète de développement en ligne ; *"Testing as a Service"* qui englobe toute la partie test et contrôle qualité du développement d'application ; *"Humain as a Service"* qui fournit l'accès à des ressources humaines proposant leurs services de prestation ; *"Gaming-as-a-Service"*, des jeux en ligne qui sont hébergés et stockés sur des serveurs dans le cloud ; etc.

1.5.2 Modèles de Déploiement

Une autre classification des services cloud consiste à considérer leurs modes de déploiement. Nous citons dans ce qui suit les quatre modèles de déploiement identifiés (voir la figure 1.5) :

Cloud privé : l'infrastructure du cloud computing est exploitée uniquement pour le compte d'une entreprise. Elle peut être interne ou externe, et gérée par l'entreprise ou un tiers.

Cloud communautaire : l'infrastructure du cloud computing est partagée entre plusieurs entreprises et s'appuie sur une communauté d'intérêts (qui partage une mission, des exigences en matière de sécurité, une politique de respect des normes, etc.). Elle peut être interne ou externe, et gérée par l'entreprise ou un tiers.

Cloud public : l'infrastructure du cloud computing est mise à la disposition du grand public et appartient à une organisation de vente de services cloud.

Cloud hybride : l'infrastructure du cloud computing est une composition de deux ou plusieurs clouds (privés, communautaires ou publics) qui demeurent des entités uniques, mais qui sont liées par une technologie standardisée ou propriétaire permettant la portabilité des données et des applications.

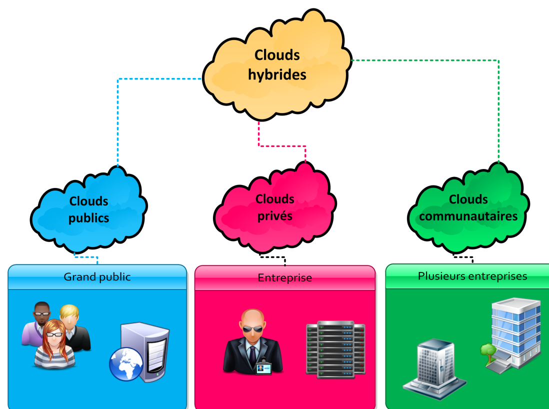


FIGURE 1.5 – Modèles de déploiement Cloud

1.6 Domaines d'usage

L'utilisation du cloud computing ne se limite pas uniquement aux entreprises à caractère commercial, elle concerne bien d'autres types d'entreprises. En fonction des raisons de sa mise en place, on distingue quatre domaines d'usage du cloud computing :

Cloud pour les Petites et Moyennes Entreprises. Nous retrouvons dans cette catégorie des entreprises de petites et moyennes tailles, disposant chacune de peu de ressources et de moyens pour l'exécution de leurs applications. L'adoption du cloud dans ce cas, offre une solution optimale pour la mutualisation des capacités de stockage et de calcul de ces entreprises.

Cloud pour la Recherche Scientifique. Pour des raisons d'accessibilité et de partage, les instituts de recherche et de développement mettent sur pied des environnements basés cloud. Ceci offre des accès exclusifs aux personnes exerçant dans le même domaine de recherche, ou appartenant aux instituts de recherche associés.

Cloud pour les Réseaux Sociaux et les Jeux. Le développement des réseaux sociaux et des jeux en ligne nécessite de plus en plus de grandes quantités de ressources. Cette nécessité est due à la croissance presque exponentielle d'utilisateurs. Dans ce contexte, l'adoption du cloud devient une évidence afin d'optimiser l'utilisation des ressources et de faciliter le partage des données vis-à-vis des utilisateurs.

Cloud pour les Fournisseurs de Services. Un fournisseur de services utilise le cloud computing pour mettre à la disposition des entreprises clientes

des plateformes exécutant leurs applications. Il s'agit d'un modèle ouvert à tout public et à caractère purement commercial.

1.7 Conclusion

Le Cloud Computing n'est pas un effet de mode comme veulent le faire croire ses détracteurs. Il représente plutôt un nouveau paradigme informatique qui a révolutionné la gestion, la distribution et le partage des ressources informatiques.

Au cours de ce premier chapitre, nous nous sommes attelés à défricher ce paradigme. Nous avons mis en revue, en particulier, ses cinq caractéristiques principales : *Libre-service à la demande, Accès étendu au réseau, Mutualisation des ressources, Élasticité/mise à l'échelle rapide et Service mesuré* ; ses trois modèles de services : *"Infrastructure as a Service", "Platform as a Service" et "Software as a Service"* ; ses quatre modèles de déploiement : *Cloud privé, communautaire, public et hybride* ; ses trois types d'utilisateurs : *administrateurs, développeurs et utilisateurs finaux* ; la notion de *virtualisation* ; ses deux scénarios de fédération : *Migration et Réplication* ; et enfin ses domaines d'usages. Malgré les avantages et les revenus potentiels qui pourraient être tirés du cloud computing, très répandu dans les entreprises commerciales, les universités, les réseaux sociaux et même les jeux vidéos, il existe encore plusieurs défis qui freinent un peu sa croissance et son adoption. Le tableau suivant 1.1 recense les principaux défis et opportunités du cloud computing [Armbrust et al., 2009].

Table 1.1 – Défis du Cloud Computing [Armbrust et al., 2009]

	Défi	Opportunité
1	"Availability of Service"	Utiliser plusieurs services issues de differents fournisseurs
2	"Data Lock-In"	Proceder à la standardisation
3	"Data Confidentiality and Auditability"	Effectuer le cryptage
4	"Data Transfer Bottlenecks"	Veiller à la sauvegarde des données
5	"Performance Unpredictability"	Améliorer la planification des machines virtuelles
6	"Scalable Storage"	Inventer un entrepôt de données évolutif
7	"Bugs in Large Distributed Systems"	Detecter les bugs dés lors de la phase de modelisation
8	"Scaling Quickly"	Pretendre le comportement dynamique des systemes cloud
9	"Reputation Fate Sharing"	Offrir des services sures
10	"Software Licensing"	Assurer le payement à l'utilisation

Chapitre 2

Approches de Modélisation des Systèmes Cloud



We can't solve problems by using the same kind of thinking we used when we created them –Albert Einstein.

Table des matières

2.1 Introduction

Les systèmes cloud, largement adoptés par les petites et moyennes entreprises, connaissent une croissance exponentielle en termes de taille et de complexité. Bien que l'émergence de ces systèmes représente la prochaine évolution dans les architectures distribuées [Marks and Lozano, 2010], il existe encore quelques défis qui freinent leur croissance et leur adoption, en particulier : "les bugs dans les grands systèmes distribués", "la disponibilité de service" et "la mise à l'échelle rapide". Plusieurs travaux existants essayent d'apporter des solutions pour remédier à certain de ces défis, chacun dans son domaine et par rapport à sa communauté. Dans le cadre de cette thèse, nous nous concentrons sur les travaux qui s'intéressent à la modélisation et l'analyse des systèmes cloud. En effet, anticiper et éviter les bugs dans un système cloud en cours d'exécution, tout en maintenant sa disponibilité, est une tâche délicate vu la complexité architecturale de ces systèmes. Étant conscient qu'un système modélisé de manière simple et claire est moins susceptible de contenir des bugs, la modélisation d'une architecture cloud peut aider non seulement à obtenir une vision plus abstraite du système, mais aussi à raisonner sur son comportement en utilisant des techniques de vérification formelle.

Dans le présent chapitre, nous classifions, étudions, et comparons les travaux existants autour de la modélisation des systèmes cloud. La classification présentée dans la section 2, se base sur l'ensemble des aspects supportés selon que le modèle décrit les aspects techniques ou théoriques d'un système cloud. Les approches à intérêt industriel (traitant les aspects techniques) sont discutées dans la section 3, alors que les approches à intérêt académique (traitant les aspects théoriques) sont présentées dans la section 4. Dans la section 5, nous comparons les différentes approches considérées afin de situer notre contribution. Enfin, la section 6 présente la conclusion.

2.2 Classification des Approches de Modélisation

Après une étude approfondie des différents travaux de modélisation existants dans la littérature, nous avons constaté qu'aux premiers stades de la recherche sur le cloud computing, il y avait seulement certaines tentatives à intérêts industriel, c'est au cours de ces dernières années, que de nouvelles approches à caractère académique ont émergé et ont stimulé le grand intérêt de la modélisation de ce type de systèmes. Partant de ce constat, nous classifions les approches de modélisation des systèmes cloud en deux grandes catégories (voir figure 2.1) ; selon le contexte considéré et les aspects supportés :

— *Travaux à intérêt industriel,* traitant les aspects techniques (technologiques ou financiers) du cloud computing ;

— *Travaux à caractère académique,* spécifiant les aspects fondamentaux (ou théoriques) des systèmes cloud.

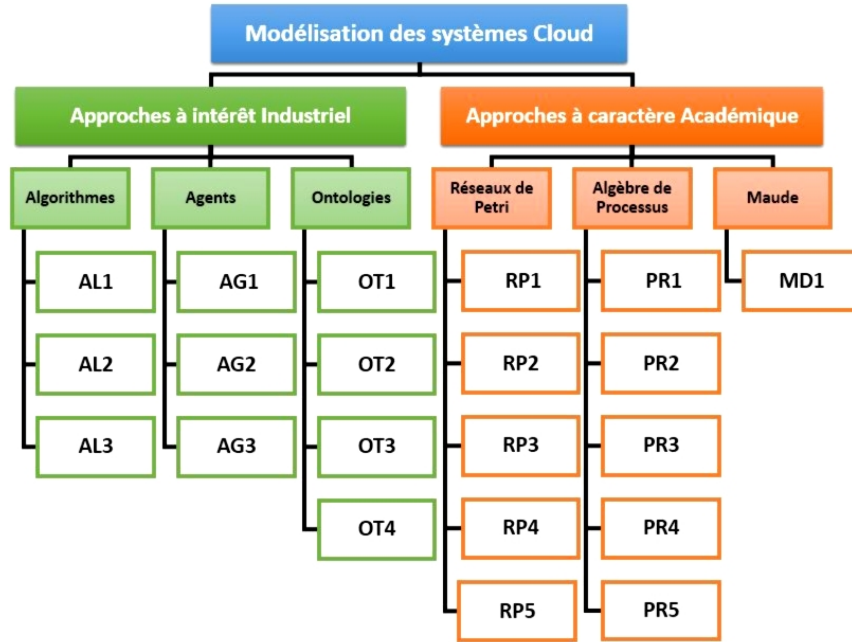


FIGURE 2.1 – Classification des approches de modélisation des systèmes cloud

Dans la première catégorie nous recensons les travaux adoptant les systèmes multi-agents (*AG1, AG2 et AG3*), les algorithmes (*AL1, AL2 et AL3*) et les ontologies (*OT1, OT2, OT3 et OT4*). Dans la deuxième catégorie, nous identifions les travaux principalement basés sur certains formalismes purement mathématiques, à savoir : les réseaux de Petri et leurs diverses extensions (*RP1, RP2, RP3, RP4 et RP5*), les algèbres de processus (*PR1, PR2, PR3, PR4 et PR5*), et enfin le langage de réécriture "Maude" (*MD1*).

2.3 Travaux à intérêt Industriel

Les travaux relatifs à cette catégorie concernent principalement la modélisation des aspects purement techniques du cloud computing. Nous identifions dans nos diverses lectures trois formalismes fortement adoptés dans ce contexte, à savoir, les systèmes multi-agents (SMA), les algorithmes et les ontologies.

2.3.1 Approches Algorithmiques

Les travaux fondés sur des algorithmes ont été largement adopté pour traiter des problèmes d'ordre économique et aussi dans un souci d'assurer la sécurité dans le cloud. Les auteurs de (AL1) [Kurdi et al., 2015] proposent un algorithme d'optimisation combinatoire pour la composition des services cloud. Cet algorithme permet de composer un nombre de services examinés avec des nuages combinés. Il garantit que le nuage avec le nombre maximal de services sera sélectionné, ce qui augmente la possibilité de satisfaire les demandes de services avec des frais généraux abordables.

Dans la même direction, mais en considérant la facturation des services cloud, les auteurs de (AL2) [Li et al., 2011] proposent un algorithme pour la mise à jour des prix lors de la planification des ressources cloud. Cet algorithme de calcul des prix (*"A Pricing Algorithm for Cloud Computing Resources"*) permet la sauvegarde des intérêts des différents participants dans les ressources cloud.

En outre, les auteurs de (AL3) [Jangra and Bala, 2013] précisent que la question de sécurité dans le cloud computing représente une des questions les plus controversées. Ils examinent trois régimes de sécurité (*"NP –No privacy", "PTP –Privacy with trusted provider" et "PNTP –Privacy with Non-Trusted Provider*) que les propriétaires de données peuvent gérer. Sur cette base, les auteurs proposent un nouvel algorithme PASA (*"Privacy Aware Security Algorithm"*) pour renforcer la protection des différents aspects de la vie privée pendant l'accès, le stockage et le traitement des données confidentielles. L'analyse de performance établie pour cet algorithme montre qu'il est très efficace et sécurisé pour préserver la confidentialité dans les systèmes cloud.

2.3.2 Approches basées Systèmes Multi-Agents

Plusieurs travaux utilisant les SMA, dont la majorités concerne principalement des aspects relatifs à la gestion des ressources dans le cloud, les processus d'affaires et la composition des services. Dans le travail de (AG1) [Nassima et al., 2014], les auteurs proposent une architecture à base d'agents pour gérer automatiquement les ressources dans le cloud. Ils proposent également un ensemble de stratégies qui aident dans le processus de prise de décision, en vue d'augmenter le profit du fournisseur. A cette fin, les auteurs mettent en avant l'objectif suivant : "*satisfaire simultanément le maximum de demandes (des ressources) acceptées avec un coût minimum et réduire autant que possible la quantité de consommation d'énergie*". Pour valider cette proposition, les auteurs réalisent une simulation en utilisant la plate-forme JADE.

Pour d'autres considérations, les auteurs de (AG2) [Lacheheub and Maamri, 2016] proposent une architecture basée sur les SMA pour la construction des processus d'affaire dans le cloud. L'architecture proposée

est constituée de deux parties : (1) "Front-end" avec des agents collectant les besoins utilisateurs, et (2) "Back-end" avec des agents construisant le processus d'affaire relatif à ces besoins. Ce travail prend en considération les besoins fonctionnels et non-fonctionnels des utilisateurs.

En outre, dans le contexte émergeant de la composition des services cloud, les auteurs de (AG3) [Gutierrez-Garcia and Sim, 2010, Gutierrez-Garcia and Sim, 2013] discutent l'idée d'implémenter une architecture multi-couches basée sur des agents pour la composition des services cloud. Ils représentent les participants et les ressources cloud via des agents "self-organizing", ensuite, ils utilisent un protocole (simulé sous JADE) pour réduire le nombre de messages entre agents afin d'augmenter la performance globale.

2.3.3 Approches basées Ontologies

Les ontologies ont été utilisées pour décrire de façon explicite la conceptualisation des connaissances représentées dans une architecture cloud. Cela permet ainsi d'apporter des solutions au problème d'hétérogénéité des termes utilisés par les différents fournisseurs cloud. Les auteurs de (OT1) [Wang et al., 2012] proposent un cadre technique basé sur l'exploitation d'un modèle d'ontologie cloud ("CBKMF –Cloud-Shadow Model Based Knowledge Service Framework") pour supporter les concepts suivants : l'incertitude, l'incohérence, la socialité et la régularité.

Dans la même direction, et après avoir discuté les pratiques actuelles et les défis de recherche concernant l'interopérabilité des entreprises, les auteurs de (OT2) [Jardim-Goncalves et al., 2013] proposent le modèle *"OEFCEI –Ontology Enriched Framework for Cloud-based Enterprise Interoperability"* enrichi par une ontologie de référence, permettant l'échange des connaissances, décisions et responsabilités au sujet des négociations. Il est validé à travers un scénario industriel.

Dans le travail proposé par (OT3) [Dastjerdi et al., 2010], les auteurs soulignent la nécessité d'une approche automatisée pour le déploiement d'applications sur les infrastructures cloud, ensuite, ils présentent une architecture basée sur les ontologies pour le déploiement d'applications cloud en fonction des qualités de services (*"QoS"*). Enfin, ils implémentent et testent leur modèle à travers les trois outils suivants : "v kernel", "rBuilder" et "VMware OVF".

En outre, les auteurs de (OT4) [Fortis et al., 2012] mettent en avant le problème de gouvernance dans le cloud et proposent une ontologie destinée à définir et analyser les aspects relatifs à la gouvernance des services cloud. Ce travail tend à consolider l'environnement de collaboration entre les différentes entreprises, en leurs fournissant les services essentiels pour la gestion de sécurité, le contrôle et l'audit.

2.4 Travaux à caractère Académique

Plusieurs travaux se sont intéressés à l'usage de formalismes assez rigoureux dans la modélisation des systèmes cloud. Ces travaux concernent en particulier les aspects purement théoriques du cloud computing. Nous présentons dans ce qui suit, les travaux les plus proches de notre contribution.

2.4.1 Approches basées Réseaux de Petri

Les réseaux de Petri et leurs extensions sont largement utilisés pour la modélisation du comportement dynamique d'un système cloud, ainsi que son analyse formelle. Comme cela a été fait dans (RP1) [Liu et al., 2013b, Fang et al., 2013], où les auteurs soulèvent l'importance du problème de la sécurité qui devient un facteur determinant dans le développement des systèmes cloud, et proposent des méthodes pour la supervision et la gestion de confiance dans le cloud. Ils procèdent par la suite à une analyse comportementale de la composition des services cloud, en se basant toujours sur les réseaux de Petri.

Dans le même sens, et afin de répondre aux préoccupations de la tolérance aux pannes de stockage d'information dans le cloud, les auteurs de (RP2) [Fitch and Xu, 2012] proposent un modèle formel pour la spécification et l'analyse des mécanismes de sécurité à l'aide des réseaux de Petri colorés (CPN). Le modèle proposé soutient non seulement le maintien de la confidentialité des données stockées, mais également assure que l'échec ou la compromission d'un fournisseur cloud individuel dans un cluster ne se traduira pas comme un compromis de l'ensemble des données.

Les auteurs de (RP3) [Zou et al., 2010], traitent la question d'assurance de la responsabilité (*"accountability"*) envers les services métiers offerts via Internet. Ils proposent un modèle de contrat entre services pour communiquer les obligations tant des consommateurs que des fournisseurs de services cloud. Ce modèle graphique (basé sur les réseaux de Petri colorés) permet aux participants de surveiller l'exécution du contrat au cours de la prestation des services, et de garder trace de l'accomplissement des obligations par chaque partie.

Un autre travail présenté dans (RP4) [Longo et al., 2011], surmonte les difficultés relatives à la dégradation des performances causée par des pannes inattendues. Les auteurs proposent par la suite, un modèle (basé sur les chaînes de Markov) pour l'analyse de la disponibilité des infrastructures-as-a-Service (IaaS). La validation de ce modèle est facilitée par une utilisation de haut niveau des réseaux de Petri stochastiques et une implémentation basée sur des scriptes Python et des modules en langage C.

Enfin, les auteurs de (RP5) [Chandramohan et al., 2012] discutent le risque de violation de la confidentialité des données utilisateurs, recueillies et conservées par les fournisseurs cloud. Par conséquent, ils proposent le modèle HPPC (*"Hierarchi-*

cal Petri-Net based Privacy nominal model approach for Cloud") pour la gestion des politiques de confidentialité dans une architecture cloud. Ce modèle permet aux utilisateurs d'avoir une confiance augmentée envers les fournisseurs cloud.

2.4.2 Approches basées Algèbre de Processus

L'algèbre de processus et ses modèles dérivés comme LOTOS (*"Language Of Temporal Ordering Specification"*), CSP ("Communicating Sequential Processes"), π-calcul , et le calcul ambiant, ont été largement adoptés pour décrire les comportements des systèmes cloud, en termes de communication et synchronisation. Ces modèles sont aussi très utiles pour la vérification de systèmes à l'aide de la logique temporelle. Les auteurs de (PR1) [Liu et al., 2013a] illustrent l'applicabilité de l'approche algébrique dans la spécification des services cloud, ils présentent une étude de cas d'un véritable système cloud industriel. L'étude de cas montre que l'approche algébrique peut identifier et éliminer les ambiguïtés et les incohérences des spécifications informelles.

De même, les auteurs de (PR2) [Kikuchi and Hiraishi, 2014] considèrent que la mauvaise configuration est la cause la plus fréquente des défaillances dans les systèmes cloud. Par conséquent, ils proposent deux techniques ; (1) synthèse automatisée des changements de configuration pour éviter les changements inappropriés et (2) identification des vulnérabilités dans la configuration du système pour augmenter la résilience d'un service en présence d'événements indésirables. Une étude de cas est aussi présentée dans ce travail afin d'évaluer le cadre formel des deux techniques.

Le travail de (PR3) [Abid et al., 2013] souligne la nécessité de protocoles pour la configuration dynamique des systèmes cloud. Les auteurs présentent un nouveau protocole (basé LNT *"LOTOS New Technology"*), capable de résoudre les dépendances dans les applications cloud fonctionnant sur différentes machines virtuelles. Ensuite, ils vérifient quelques propriétés du protocole proposé au moyen d'outils disponibles dans le "CADP toolbox".

En outre, les auteurs de (PR4) [Rezaee et al., 2014] introduisent le concept de FICS (*"Fuzzy Inference Cloud Service"*) et proposent une nouvelle approche pour la modélisation formelle de FICS. Ils présentent également un ensemble de propriétés de vérification pour garantir la cohérence interne et l'absence d'interblocage du FICS.

En utilisant le π-calcul, les auteurs de (PR5) [YANG and WANG, 2012] proposent un modèle de sécurité pour représenter formellement les actions de négociation des différents participants. Ils procèdent également à la vérification de l'exactitude des combinaisons de services cloud.

2.4.3 Approche basée Maude

Basé sur la logique de réécriture, le langage formel "Maude" représente un des meilleurs langages dans le domaine de la modélisation des systèmes concurrents [Clavel et al., 2007]. Il vise à maximiser la simplicité, l'expressivité et la performance [Clavel et al., 2007]. La logique de réécriture, dotée du langage formel "Maude", a été étroitement adoptée dans notre contexte. Les auteurs de (MD1) [Muehlbauer, 2012] proposent une solution formelle basée sur "Maude" afin de garantir le bon fonctionnement des systèmes de gestion du cloud computing. Ils proposent le langage "D-KLAIM" pour la spécification *algébrique* de ces systèmes, dont la structure statique est décrite par des équations, et le comportement dynamique est spécifié via la notion de règles de réécriture. Les auteurs discutent également la mise en œuvre de leur proposition à travers l'analyse des propriétés relatives à l'exclusion mutuelle et la vivacité.

2.5 Comparaison et Synthèse

Nous procédons dans cette section à la comparaison des travaux présentés dans les sections précédentes, pour se faire, nous identifions trois critères de comparaison ; jugés les plus pertinents par rapport à notre étude.

Le premier critère concerne le degré d'expressivité du modèle proposé (la modélisation). Il est défini par rapport au formalisme utilisé et aux aspects cloud supportés. Nous identifions six aspects essentiels pour la spécification d'architecture cloud (voir tableau 2.1), issues de la définition 1.4 suggérée dans le chapitre 1.

Le deuxième critère adresse l'analyse du modèle proposé par rapport à l'ensemble des propriétés spécifiées. Il concerne principalement la technique utilisée et la nature des propriétés souhaitées.

Le troisième critère s'intéresse à la mise en œuvre des résultats théoriques obtenus via une implémentation appropriée, ou un outil dédié.

TABLE 2.1 – Aspects d'expressivité

e1	IU	Interactions des utilisateurs dans le cloud
e2	CS	Composition des services cloud
e3	CV	Communication entre machines virtuelles
e4	EU	Emplacement des utilisateurs dans le cloud
e5	LS	Localité des services cloud
e6	LV	Localisation des machines virtuelles

Table 2.2 – Comparaison des approches de modélisation

	Modélisation							**Analyse**		**Mise en œuvre**
Travaux	***Formalisme***	***Aspects supportés*** ***e1***	***e2***	***e3***	***e4***	***e5***	***e6***	***Technique utilisée***	***Propriétés souhaitées***	
AG1	Agents	+	-	+	-	-	-	Simulation	Augmentation du profit des fournisseurs et Minimisation des rejets de demandes des machines virtuels.	Plateforme JADE.
AG2		+	+	-	-	-	-	/	/	/
AG3		+	+	-	-	-	-	Simulation	Composition des services cloud, de manière autonome.	Plateforme JADE.
ALl	Algorithmes	+	+	-	-	-	-	Simulation	Satisfaire les demandes de services avec des frais généraux minimes.	Implémentation sous JAVA.
AL2		+	-	-	-	-	-	Simulation	Sauvegarde des intérêts des participants dans les ressources cloud.	Implémentation.
AL3		+	-	-	-	-	-	Simulation	Confidentialité, Authentification, Intégrité et Performance contre les attaques.	Implémentation.
OT1	Ontologies	+	-	-	-	-	-	Validation	Incertitude, Incohérence, Socialité et Régularité.	Implémentation.
OT2		+	-	+	-	-	-	Validation	Interopérabilité.	
OT3		+	-	+	-	+	-	Execution	Efficience et Efficacité.	Implémentation via vkernel, rBuilder et VMware OVF Tool.
OT4		+	-	-	-	+	-	Validation	Consolider la gouvernance dans le cloud.	Implémentation.
RP1	Réseaux de Petri	-	+	-	-	-	-	Vérification	Satisfiabilité fonctionnelle et Prévisibilité du comportement.	Outils existants.
RP2		+	-	-	-	+	-	Vérification	Vivant, Bornée, et Sans-blocage.	Outils existants.
RP3		+	+	-	-	-	-	Validation	Exactitude.	Outils existants.
RP4		-	+	-	-	-	-	Simulation	Haute disponibilité des IaaS.	Implémentation via Python,et le langage C.
RP5		+	-	-	-	+	-	Vérification	Preservation de la vie privée des utilisateurs	Outils existants.
PR1	Algèbre de Processus	+	-	-	-	+	-	Vérification	Cohérence et reduction de la Redondance.	Implémentation via le langage CASOCC-WS.
PR2		-	-	-	-	+	+	Vérification	Fiabilité et Disponibilité des services cloud.	Implémentation via Solver-tool, Alloy Analyzer's et NuSMV.
PR3		-	-	+	-	+	-	Vérification	Résoudre les dépendances entre les machines virtuels.	Implémentation via JAVA et ADP toolbox.
PR4		-	+	-	-	-	-	Vérification	Cohérence, Inter-blocage, Divergence et Accessibilité.	Outils existants.
PR5		-	+	-	+	-	-	Vérification	Exactitude de combinaison de services.	Outils existants.
MD1	Maude	+	-	+	-	+	-	Vérification	Exclusion mutuelle et Vivacité	Implémentation basée Maude.

D'après l'étude approfondie des différents travaux existants –dont les résultats de comparaison sont présentés dans le tableau 2.2, nous avons dégagé les constatations suivantes :

— Les travaux à intérêt industriel proposent des solutions *spécifiques* pour traiter un problème d'ordre technologiques ou financiers ; souvent imposé par un fournisseur ou un utilisateur. Dans cette lignée, les travaux basés sur les systèmes multi-agents concernent principalement que des questions relatives à la gestion des ressources dans le cloud, ceux basés algorithmique se montrent plus appropriées pour assurer l'intégrité de la découverte et la composition des services cloud. Aussi, les travaux orientés ontologies sont particulièrement adaptés à la gestion de l'interopérabilité entre les différents fournisseurs cloud.

— Les travaux à caractère académique proposent des solutions *abstraites mais non-génériques* ; les formalismes utilisés ne sont pas capables d'exprimer autant d'aspects théoriques dans les architectures cloud. Dans cette catégorie d'approches, les travaux utilisant les réseaux de Petri reposent sur la théorie comportementale pour la description des interactions existantes entre les utilisateurs et les services cloud. De même, les approches basées sur l'algèbre de processus et celle basée Maude, s'intéressent à l'aspect interaction mais en termes de communication et synchronisation.

— D'une manière générale, et en terme d'expressivité (*premier critère*), nous constatons que la spécification des aspects relatifs à la virtualisation (*e3 et e6*), ont été occultés par rapport aux aspects relatifs aux utilisateurs (*e1 et e4*) et services cloud (*e2 et e5*). D'autre part, nous remarquons aussi que très peu de concentration a été orientée vers les aspects de localité (*e4, e5 et e6*), contrairement aux aspects d'interaction (*e1, e2 et e3*).

— En terme d'analyse et mise en œuvre des modèles théoriques (*deuxième et troisième critère*), nous notons que les travaux à intérêt industriel utilisent le plus souvent la *simulation* et la *validation* via des implémentations. Tandis que la *vérification*, est particulièrement utilisée dans les travaux à caractère académique via des outils appropriés. En outre, la nature des propriétés considérées dans les deux catégories de travaux, est de très bas niveau et concerne des propriétés standards (telles que : l'exclusion mutuelle, la vivacité, l'exactitude, etc.) ; à l'exception de quelques travaux.

Il s'avère aussi que la modélisation d'une architecture cloud, en considérant toutes ses facettes, n'est pas un travail aussi simple, telle que pour une architecture distribuée ordinaire. L'ensemble des travaux évoqués représentent des solutions très restrictives dans la description des aspects fondamentaux du cloud. La solution idéale consiste donc à adopter un formalisme capable de fournir des modèles étant suffisamment expressifs tout en restant raisonnablement analysables. Les bigraphes [Milner, 2009] représentent une des théories les plus utilisées pour la spécification

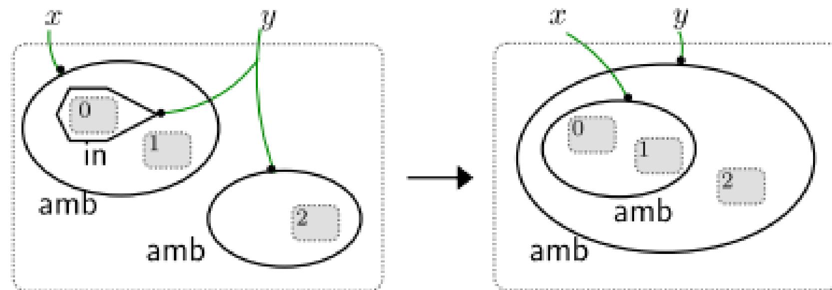
des systèmes ubiquitaires, dans ce travail, nous les adoptons pour la première fois [Benzadri et al., 2013a] pour modéliser les systèmes cloud. Ce choix est motivé notamment par leur degré d'expressivité ; un bigraphe supporte deux vues différentes du système modélisé : *(i) distribution spatial* et *(ii) connectivité*, en plus de la possibilité de décrire son comportement dynamique. Les bigraphes sont également dotés d'une représentation graphique et une spécification algébrique équivalente.

2.6 Conclusion

Dans ce chapitre, nous avons réalisé une étude comparative entre les approches de modélisation des systèmes cloud. Cette étude adopte trois critères de comparaison : *(i) modélisation* –determinant le degré d'expressivité du modèle proposé, *(ii) analyse* –recensant l'ensemble de propriétés vérifiées et *(iii) mise en œuvre* –décrivant la concrétisation du modèle proposé à travers une implémentation ou utilisation des outils existants. Nous avons mis en évidence *vingt et un travaux* classés en deux catégories (travaux à intérêt industriel et travaux à caractère académique), et adoptant différents formalismes (*Agents, Algorithmes, Ontologies, Réseaux de Petri, Algèbre de processus et Maude*). Le principal constat qui découle de cette étude est l'absence cruciale de l'adoption d'un formalisme capable de prendre en compte les éléments de base d'une architecture cloud, vue selon toutes ses facettes.

Chapitre 3

Théorie des Bigraphes



"We are mathematicians, and it is for those in computing science, not us, to determine which is the best model for a given application"
–Barre and Walls.

Table des matières

3.1 Introduction

Au même titre que les mathématiques, les notions fondamentales de l'informatique entretiennent des relations profondes avec la théorie des catégories ; en tant que langage formel appliqué à la modélisation des systèmes complexes [Barre and Wells, 1990]. L'utilisation d'un langage formel offre un haut niveau d'abstraction permettant d'exposer des énoncés de manière précise et sans ambiguïté ; en proposant des modèles formels suffisamment expressifs et mathématiquement analysables. Plusieurs formalismes ont été principalement basés sur la théorie des catégories, parmi eux, les bigraphes introduits par [Milner, 2009] afin de modéliser les systèmes ubiquitaires. Un bigraphe prend en charge deux dimensions différentes dans la même structure : *distribution spatiale* et *interaction*, du système à spécifier. Alors que la première dimension concerne en particulier la spécification de la localisation des entités physiques ou logicielles, la deuxième représente les connectivités possibles entre ces différentes entités. De plus, ce formalisme offre la possibilité de modéliser l'évolution dynamique d'un système à l'aide d'un ensemble de règles de réaction. Enfin, il convient notamment de noter qu'un bigraphe est doté d'une représentation graphique et une spécification algébrique équivalente.

Ce chapitre présente les concepts fondamentaux de la théorie des bigraphes, en insistant sur les notions qui seront exploitées lors de la présentation de notre contribution. Dans la section 2, nous décrivons informellement les notations graphiques représentant un bigraphe et nous donnons ensuite les définitions formelles de ses constituants, à savoir le graphe de places et le graphe de liens. La section 3 est consacrée à la définition des termes algébriques et des opérations fondamentales sur les bigraphes, en particulier la composition et la juxtaposition. Aussi, la discipline de typage des bigraphes est décrite dans la section 4. La section 5 traite la dynamique des bigraphes à travers les règles de réaction. Quelques extensions et applications du formalisme de base (les bigraphes) sont données dans la section 6. L'ensemble des outils pratiques d'édition et d'exécution des bigraphes, est présenté dans la section 7.

3.2 Constituants d'un Bigraphe

Un bigraphe, tel qu'un graphe ordinaire, est composé de nœuds et d'arêtes. La figure 3.1 illustre un bigraphe sous sa forme graphique. Les rectangles en pointillé indiquent les racines ("Roots") ; elles décrivent les parties adjacentes du système. Les cercles sont les nœuds du bigraphe ; leur emplacement spatial est décrit par leur imbrication. Les carrés gris sont appelés des sites, ils représentent des empla-

cements logiques dans lesquels une racine ou un nœud peut être inséré. L'ensemble des nœuds, sites et racines représente les places d'un bigraphe. Les interactions entre les différents nœuds sont représentées par des hyper-arcs désignés par *ei*. Les points de liaison entre les nœuds et les arcs sont appelés ports. Un contrôle (type) est attaché à chaque nœud du bigraphe ; il indique le nombre de ports qu'un nœud peut avoir et permet de déterminer s'il est atomique (nœud vide), actif (nœud permettant l'application de règles de réaction à l'intérieur) ou bien passif. La signature d'un bigraphe prend la forme (S, ar) ; avec S étant l'ensemble des contrôles sur les nœuds (types), et $ar : S \rightarrow N$, est une fonction attribuant un nombre de ports (arité) à chaque type de nœud. Un bigraphe peut aussi avoir des noms internes et des noms externes (*"inner-names"* et *"outer-names"*) ; Ils précisent les liens potentiels avec d'autres bigraphes.

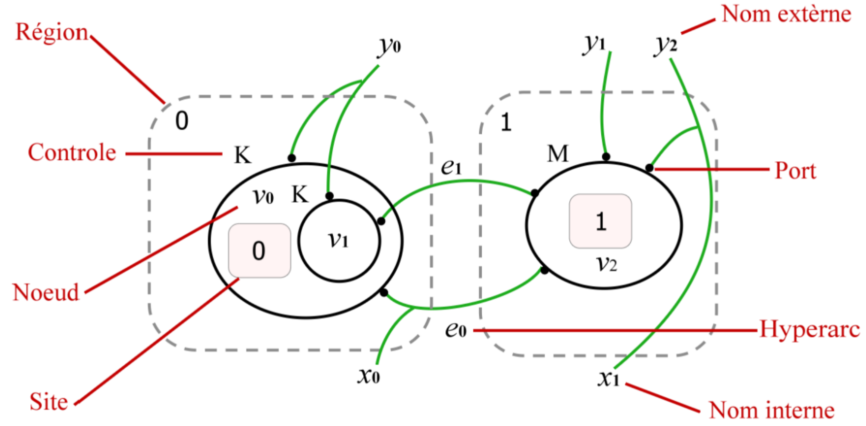


FIGURE 3.1 – Anatomie des Bigraphes [Cherfia, 2016]

Le bigraphe comme son nom l'indique (*bi*-graphe), est constitué de deux graphes indépendants : le graphe de places ; exprimant la localité physique des nœuds, et le graphe de liens ; représentant les connectivités entre ces nœuds. La définition formelle (3.1) d'un bigraphe, telle que introduite par [Milner, 2009], est comme suit :

Définition 3.1 ***Bigraphe***
Un bigraphe est le 5-uplet : $G = (V, E, ctrl, prnt, link) : I \rightarrow J$*, également écrit :* $< GP, GL >$*, où :*

- $GP = (V, ctrl, prnt) : m \rightarrow n$*, est le graphe de places,*
- $GL = (V, E, ctrl, link) : X \rightarrow Y$*, est le graphe de liens,*
- *I, J sont respectivement les interfaces entrantes* $(I =< m, X >)$ *et sortantes* $(J =< n, Y >)$ *du bigraphe.*

L'intersection entre ces deux graphes (graphe de places et graphe de liens) est l'ensemble commun de nœuds (V). Alors qu'une arête dans le graphe de places montre la relation d'imbrication entre les nœuds du bigraphe, un hyper-arc dans le graphe de liens établit une connexion entre les ports de ces nœuds.

3.2.1 Graphe de Places

Le graphe de places permet la spécification de la notion de localité (imbrication de nœuds) dans un bigraphe. Il est constitué d'une forêt composée de plusieurs arbres (comme l'illustre la figure 3.2), dont la racine est toujours une région à laquelle sont rattachées hiérarchiquement plusieurs feuilles (des nœuds ou des sites). La figure 3.2 décrit le graphe de places du bigraphe présenté dans la figure 3.1.

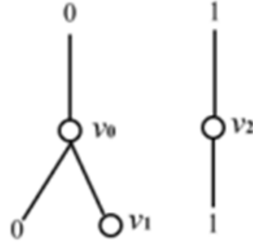


FIGURE 3.2 – Graphe de Places

Dans le graphe de places, la hiérarchie des nœuds reflète une relation père/fils; montrant ainsi que les nœuds internes sont des fils des nœuds externes. Dans ce qui suit, nous détaillerons la définition formelle (3.2) d'un graphe de places.

Définition 3.2 ***Graphe de Places***
Un graphe de places est le 3-uplet : $GP = (V, ctrl, prnt) : m \to n$*, où :*
- *V est un ensemble fini de nœuds,*
- *$ctrl : V \to S$ est une transformation assignant un contrôle à chaque nœud; la signature S est un ensemble dont les éléments sont appelés des contrôles (types de nœuds),*
- *$prnt : m \uplus V \to V \uplus n$ est une transformation indiquant le nœud parent,*
- *m est le nombre de sites,*
- *n est le nombre de régions.*

Les interfaces externes et internes d'un graphe de places, représentent respectivement le nombre de régions (racines) et le nombre de sites.

3.2.2 Graphe de Liens

Un graphe de liens permet de décrire la notion de connectivité (interaction entre les nœuds) d'un bigraphe. La figure 3.3 décrit le graphe de liens du bigraphe présenté dans la figure 3.1.

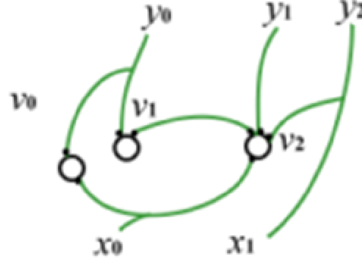


FIGURE 3.3 – Graphe de Liens

Le graphe de liens est un hyper-graphe, dans lequel chaque arc sert à relier plusieurs nœuds à la fois. Ces arêtes sont aussi des liens non orientés ; un lien de A vers B ou inversement (B vers A) représente la même arête. Dans ce qui suit, nous présentons la définition formelle d'un graphe de liens [Milner, 2009].

Définition 3.3 ***Graphe de Liens***
Un graphe de liens est le 3-uplet : $GL = (V, E, ctrl, link) : X \to Y$*, où :*

- V *est un ensemble fini de nœuds,*
- E *est un ensemble d'arêtes,*
- $ctrl : V \to S$ *est la transformation de contrôle des nœuds,*
- $link : X \uplus P \to E \uplus Y$ *est la transformation de liaison entre les ports des nœuds et les arêtes du bigraphe,*
- X *est l'ensemble des noms internes,*
- Y *est l'ensemble des noms externes.*

Le graphe de liens comporte des interfaces d'entrées (X) et des interfaces de sorties (Y), représentant des points de connections avec d'autres graphes de liens (son environnement extérieur).

Exemple 1

L'exemple suivant (voir figure 3.4) [Milner, 2009], montre un simple bigraphe modélisant cinq utilisateurs effectuant une visio-conférence. La signature associée à cet exemple est $S = \{A : 2, B : 1, C : 2, R : 0\}$; définissant les différents types de nœuds du bigraphe : utilisateurs (A), bâtiments (B), ordinateurs (C) et salles

(R). Dans une salle, chaque utilisateur peut se connecter à travers un ordinateur, qui est relié aux autres ordinateurs du bâtiment via un réseau local.

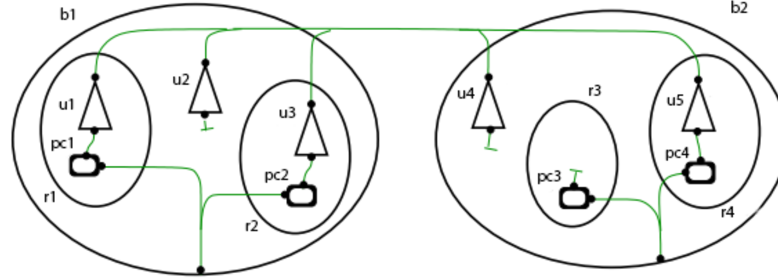


FIGURE 3.4 – Bigraphe modélisant une visio-conférence [Milner, 2009]

L'ensemble des nœuds de ce bigraphe est défini par V = {*u1, u2, u3, u4, u5, pc1, pc2, pc3, pc4, r1, r2, r3, r4, b1, b2*}. Alors que l'ensemble des arcs est donné par $E = \{e0, e1, e2, e3, e4, e5\}$; il désigne le réseau d'interconnexion entre les utilisateurs. La transformation de contrôle (ctrl) renvoie dans notre cas le résultat suivant : S = {*(u1 : A), (u2 : A), (u3 : A), (u4 : A), (u5 : A), (pc1 : C), (pc2 : C), (pc3 : C), (pc4 : C), (r1 : R), (r2 : R), (r3 : R), (r4 : R), (b1 : B),(b2 : B)* }. On peut également écrire le bigraphe G spécifiant cet exemple comme suit : $G =< 0, \phi > \to < 2, x >$; son interface d'entrée indique qu'aucun site n'est disponible (m=0), alors que son interface de sortie précise qu'il existe deux régions (n=2).

3.3 Opérations Algébriques sur les Bigraphes

En plus de leur représentation graphique (figure 3.1), les bigraphes sont dotés d'une spécification algébrique. Le tableau3.1 résume les termes algébriques de base du langage bigraphique.

TABLE 3.1 – Termes Algébriques des Bigraphes

Terme	**Signification**
Sa(U)	Nœud U avec le contrôle S d'arité a.
U.V	Nœud U contient le nœud V.
x/y	Connexion du nom interne y au nom externe x.
$U \mid V$	Produit principal (Juxtaposition de nœuds).
$U \parallel V$	Produit parallèle (Juxtaposition des racines).
$U \otimes V$	Produit tensoriel.
$U \circ V$	Composition.

Il convient aussi de noter qu'on peut constituer des bigraphes plus complexes à partir de bigraphes plus simples, grâce à un ensemble d'opérations algébriques. Milner [Milner, 2009] définit deux types d'opérations sur les bigraphes ; horizontale appelée *"Produit Tensoriel"* (ou Juxtaposition), et verticale dite simplement *"Composition"*. Les sous-sections suivantes sont réservées à la présentation détaillée des deux opérations.

3.3.1 Produit Tensoriel

Cette opération n'est définie qu'entre deux bigraphes ayant des interfaces disjointes. Le produit tensoriel consiste à mettre côte à côte les graphes de places des bigraphes à composer, construire l'union de leurs graphes de liens (en joignant les arcs ouverts correspondants), et enfin, augmenter le nombre de sites d'un bigraphe par le nombre de sites de l'autre. La définition formelle de cette opération est donnée comme suit [Milner, 2009] :

Définition 3.4 ***Produit Tensoriel*** *Le produit tensoriel ($F_0 \otimes F_1$) des bigraphes $F_i; i \in 0,1$ est défini séparément en termes de graphes de places $FP_i; i \in 0,1$ et graphes de liens $FL_i; i \in 0,1$:*

- *Si $FP_i = (V_i, ctrl_i, prnt_i) : m_i \to n_i/(i = 0,1)$ sont des graphes de places d'interfaces disjointes, le produit tensoriel $FP_0 \otimes FP_1 : m_0 + m_1 \to n_0 + n_1$ est donné par :*
  $FP_0 \otimes FP_1 = (V_0 \uplus V_1, ctrl_0 \uplus ctrl_1, prnt_0 \uplus prnt_1)$ *, dont :* $prnt_1(m_0 + i) = n_0 + j$ *à chaque fois que* $prnt_1(i) = j$*.*

- *Si $FL_i = (V_i, E_i, ctrl_i, prnt_i) : X_i \to Y_i$ sont des graphes de liens avec des interfaces disjointes (i =0, 1), le produit tensoriel $FL_0 \otimes FL_1 : X_0 \uplus X_1 \to Y_0 \uplus Y_1$ est donné par :*
  $FL_0 \otimes FL_1 = (V_0 \uplus V_1, E_0 \uplus E_1, ctrl_0 \uplus ctrl_1, link_0 \uplus link_1)$ *.*

- *Si $F_i :< m_i, X_i > \to < n_i, Y_i >$ sont des bigraphes avec des interfaces disjointes (i = 0, 1), le produit tensoriel $F_0 \otimes F_1 :< m_0 + m_1, X_0 \uplus X_1 > \to < n_0 + n_1, Y_0 \uplus Y_1 >$ est donné par :*
  $F_0 \otimes F_1 =< FP_0 \otimes FP_1, FL_0 \otimes FL_1 >$.

Exemple 2

Le produit tensoriel des deux bigraphes F_0 et F_1 (voir la figure 3.5), possédant respectivement comme interfaces les noms externes :$\{x, w\}et\{y, w\}$, est donné par le bigraphe suivant : $F = F_0 \otimes F_1$ avec l'ensemble des noms externes : $Y_0 \uplus Y_1 = \{x, y, w\}$; en reliant les arcs ouverts ayant le même nom (w) [Benlahrache, 2014].

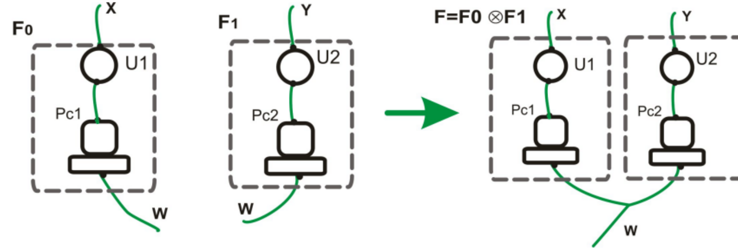


FIGURE 3.5 – Exemple de Produit Tensoriel [Benlahrache, 2014]

3.3.2 Composition

La composition est une opération plus complexe que le produit tensoriel, elle peut être considérée comme une imbrication d'un bigraphe à l'intérieur d'un autre. Cette opération permet donc une construction modulaire dans laquelle les régions d'un bigraphe sont hébergées dans les sites du bigraphe hôte. Ainsi, il est nécessaire que les interfaces externes du bigraphe à héberger correspondent aux interfaces internes du bigraphe hôte. La composition de deux bigraphes est définie séparément pour la composition des graphes de places et des graphes de liens ; comme le précise la définition 3.5 suivante[Milner, 2009] :

Définition 3.5 ***Composition des Bigraphes***

La composition des bigraphes ($F \otimes G$) est définie comme suit :

- *Si $F_P : k \to m$ et $G_P : m \to n$ sont deux graphes de places, leurs composition $G_P \circ F_P = (V, ctrl, prnt) : k \to n$, dont l'ensemble de nœuds $V = V_F \uplus V_G$ et la transformation de contrôle $ctrl = ctrl_F \uplus ctrl_G$. Aussi la transformation de parenté (prnt) est définie comme suit : Si $w \in k \uplus V_F \uplus V_G$ est un site ou un nœud de $G_P \circ F_P$ alors prnt(w) =*
  - *$prnt_F(w) si w \in k \uplus V_F et prnt_F(w) \in V_F$;*
  - *$prnt_G(j) si w \in k \uplus V_F et prnt_F(w) = j \in m$;*
  - *$prnt_G(w) si w \in V_G$.*
- *Si $F_L : X \to Y$ et $G_L : Y \to Z$ sont deux graphes de liens, leur composition $G_L \circ F_L = (V, E, ctrl, link) : X \to Z$, dont l'ensemble de nœuds $V = V_F \uplus V_G, E = E_F \uplus E_G, ctrl = ctrl_F \uplus ctrl_G$, et la transformation de liaison (link) est définie comme suit : Si $q \in X \uplus P_F \uplus P_G$ est un port de $G_L \circ F_L$ alors link(q) =*
  - *$link_F(q) si q \in X \uplus P_F et link_F(q) \in E_F$;*
  - *$link_G(y) si q \in X \uplus P_F et link_F(q) = y \in Y$;*
  - *$link_G(q) si q \in P_G$.*
- *Si $F : I \to J$ et $G : J \to K$ sont deux bigraphes, leur composition est $G \circ F =< G_P \circ F_P, G_L \circ F_L >: I \to K$.*

Exemple 3

La composition du bigraphe F constitué d'une région ayant (x et y) comme noms externes et le bigraphe H possédant un site et deux noms internes (x et y), est le bigraphe résultant : $G = H \circ F$ sans sites ni noms externes (voir la figure 3.6). De ce fait, les éléments du bigraphe F sont intégrés dans les éléments du bigraphe H à travers le site (0) [Benlahrache, 2014].

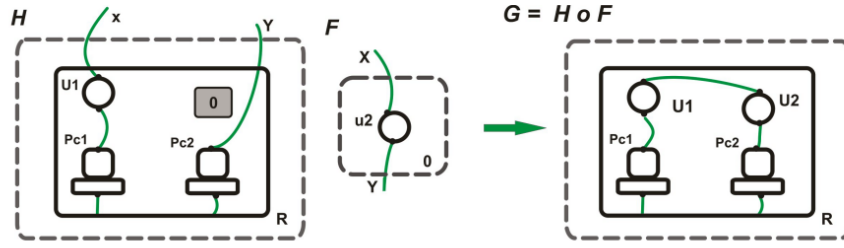


FIGURE 3.6 – Exemple de Composition [Benlahrache, 2014]

Axiomes des Opérations

Les axiomes de base de la théorie des bigraphes sont principalement issus de la définition des opérations de composition et du produit tensoriel. Le rôle de ces axiomes est de formaliser l'associativité des deux opérations, et la distribution du produit tensoriel sur la composition [Sevegnani, 2012a].

— $A \circ id_X = A = id_Y \circ A$ avec $A : X \to Y$,
— $A \circ (B \circ C) = (A \circ B) \circ C$,
— $A \otimes (B \otimes C) = (A \otimes B) \otimes C$,
— $(A0 \otimes B0) \circ (A1 \otimes B1) = (A0 \circ A1) \otimes (B0 \circ B1)$.

Il convient de noter que les identités (notés par *'id'*) sont des bigraphes élémentaires sans nœuds, qui constituent l'élément neutre de l'opération de composition [Sevegnani, 2012a].

3.4 Typage des Places et des Liens

La discipline de typage a été définie par Milner [Milner, 2009] pour imposer un ensemble de contraintes sur la structure d'un bigraphe. Une telle discipline spécifie une signature plus riche via deux types de contraintes : nous pouvons contraindre les places (voir définition 3.6) ou bien les liens (voir définition 3.7) d'un bigraphe. Le typage de places ("place-sorting") permet de contraindre les fils d'un nœud

d'avoir seulement certains contrôles, tandis que le typage de liens ("link-sorting") permet de contraindre la liaison permise entre les ports des différents nœuds. Cela garantit que seuls les bigraphes avec une représentation précise sont ainsi formulés.

Définition 3.6 ***Typage de places***
Un typage de places $\sigma^P = (\Theta^P, S, \Phi^P)$ *possède un ensemble non-vide* Θ^P *de sortes. La signature* S *attribue une sorte à chaque contrôle. La composante* Φ^P *représente l'ensemble de règles de formation de places.*

De la même manière, le typage de liens est définie comme suit :

Définition 3.7 ***Typage de liens***
Un typage de liens est le triplet $\sigma^L = (\Theta^L, S, \Phi^L)$*, où* Θ^L *est un ensemble non vide de sortes et* S *est une signature qui attribue une sorte à chaque port d'un contrôle donné. La composante* Φ^L *est l'ensemble de règles de formation des liens.*

3.5 Systèmes Réactifs Bigraphiques

La structure d'un bigraphe prend en charge deux aspects essentiels du système à spécifier : la localité et la connectivité des nœuds. Pour modéliser sa dynamique, la théorie des bigraphes offre la notion de "BRS" (Systèmes Réactifs Bigraphiques). Un BRS est constitué d'un ensemble de bigraphes représentant l'état du système, et d'un ensemble de règles de réaction définissant la façon dont le système peut évoluer. La définition formelle d'une règle de réaction est la suivante [Milner, 2009] :

Définition 3.8 ***Règle de réaction***
Une règle de réaction prend la forme (R, R', O), où :
- *R :* $m \to n$*, est le bigraphe à transformer, appelé "redex",*
- *R' :* $m' \to n$*, est le bigraphe résultant de la transformation, il est appelé "reactum",*
- *O :* $m' \to m$*, est une fonction d'ordinaux établissant la correspondance entre les interfaces internes de R' et R.*

L'application d'une règle de réaction repose sur le fait d'identifier dans le bigraphe contextuel une copie du bigraphe "redex" puis la remplacer par le bigraphe "reactum". Elle peut affecter une transformation sur les places ; représentant en général le déplacement d'un nœud, ou bien une transformation sur les liens ; exprimant la connexion ou déconnexion d'un nœud.

Exemple 4

Un exemple d'application de trois règles de réaction bigraphiques est représenté dans la figure 3.7. La première règle (R1) spécifie qu'un utilisateur peut quitter une visio-conférence, alors que la deuxième règle (R2) représente une connexion à un ordinateur dans le même endroit que son utilisateur (probablement une salle). Ces deux règles (R1 et R2) agissent sur les liaisons uniquement et non pas les places d'un bigraphe. Enfin, la règle R3 exprime la possibilité qu'un utilisateur entre dans une salle, elle ne modifie que les places du bigraphe. Le site (nœud ombré) permet à la salle de contenir d'autres nœuds, par exemple un ordinateur ou d'autres utilisateurs.

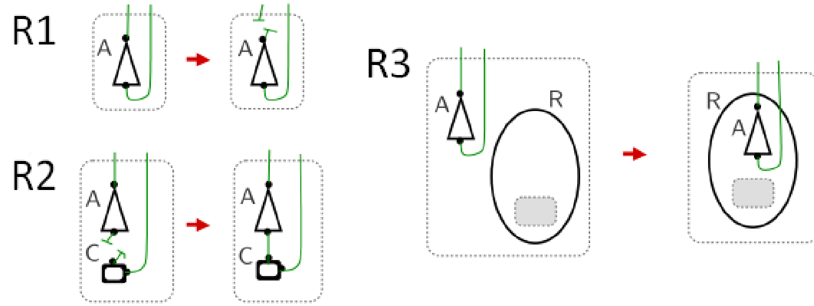


FIGURE 3.7 – Exemple d'application d'une règle de réaction [Milner, 2009]

3.6 Extensions des Bigraphes

Depuis leur première introduction en 2004 [Milner and Jensen, 2004], plusieurs extensions et raffinements ont été rajoutés à la définition standard des bigraphes. Quatre extensions principales ont été réalisées sur les bigraphes dans lesquelles de nouvelles notions ont ainsi été introduites : les contraintes sur la localité des arcs dans les graphes de liens, l'ajout d'une probabilité aux arcs donnant lieu aux bigraphes stochastiques, l'ajout d'arcs orientés dans les graphes de liens, et enfin, les bigraphes introduisant la possibilité qu'un nœud puisse hériter de plusieurs nœuds parents. Dans ce qui suit, nous présentons de façon assez détaillée ces quatre extensions ;

"Binding Bigraphs". Dans cette extension des bigraphes, la séparation entre les deux notions de localité et connectivité est assouplie via l'assignement d'une localité à quelques arêtes d'un bigraphe. Ceci, est uniformisé via l'autorisation de mettre les noms d'un bigraphe au sein des racines et sites. Par conséquent, il devient important de savoir si une arête franchit les limites d'une place. Par exemple, une arête assignée à

un nœud ne peut relier que les ports situés au sein de ce nœud. Les structures définies par cette extension ont été principalement formalisées dans [Damgaard and Birkedal, 2005]. En particulier, les intérêts de ces bigraphes ont été prouvés dans la modélisation des protocoles de sécurité privatisant la communication dans certains lieux uniquement.

"Stochastic Bigraphs". Cette deuxième extension des bigraphes a été principalement présentée dans[Krivine et al., 2008]. Elle consiste à associer une valeur stochastique aux règles de réaction. De ce fait, l'espace d'état généré par cette extension peut être interprété en une chaîne de markov à temps continu (CTMC). Sur cette base, et vu qu'une réaction peut se produire en appliquant différentes règles de réaction sous-jacentes, l'apport de chaque règle doit être pris en considération. Par conséquent, le taux de réaction pour une règle est obtenu par le produit du taux constant et le nombre d'occurrences de la règle. Un tel taux représente le paramètre d'une distribution exponentielle caractérisant le comportement stochastique de cette réaction. En particulier, l'expressivité des bigraphes stochastiques a été illustrée dans le contexte des systèmes moléculaires utilisant des compartiments, ainsi que dans l'analyse stochastique des systèmes distribués de manière générale.

"Directed Bigraphs". Les auteurs des bigraphes dirigés [Grohmann and Miculan, 2007], proposent une nouvelle définition des graphes de liens, dans laquelle les arêtes sont considérées comme des ressources et les noms sont interprétés comme des demandes de ces ressources. Ainsi, la principale nouveauté dans cette extension est d'attribuer une direction (un sens) aux arêtes afin de capturer le flux de demande des ressources. Dans cette extension des bigraphes, les graphes de liens orientés étaient représentés avec la particularité que les arêtes sont explicitement représentées comme des sommets d'un graphe, et non pas comme des hyper-arêtes reliant les noms et les ports. Les bigraphes dirigés peuvent être utilisés comme une théorie générale pour fournir des systèmes de transition étiquetés à partir des systèmes réactifs. Aussi, ces bigraphes ont montré leurs intérêts dans le domaine de la sécurité des systèmes informatiques (la cryptographie).

"Bigraphs with sharing" Cette extension représente une généralisation des bigraphes standards de Milner dans laquelle les places peuvent avoir plusieurs parents. Une version préliminaire de ce travail a été présentée dans [Sevegnani and Calder, 2015]. L'introduction du partage dans les bigraphes est justifiée par le fait que la localisation basée sur "DAG" (graphe orienté acyclique) est souvent plus naturelle lors de la modélisation du monde réel. Ceci est formalisé dans les bigraphes en maintenant une copie d'un nœud partagé à l'intérieur de chacun de ses parents, ensuite, relier toutes les copies par un lien spécial.

3.7 Outils pratiques autour des Bigraphes

Une des principales idées des bigraphes réside dans la modélisation des systèmes distribués. Cependant, la manipulation et l'analyse manuelle de ces modèles est d'une extrême complexité. Par conséquent, le développement d'outils logiciels est essentielle pour déterminer si les modèles bigraphiques sont appropriés pour une future utilisation dans des situations réelles. Selon son fondateur Milner [Milner, 2009], l'implémentation d'outils pour les bigraphes concerne en particulier les trois points suivants : (i) la programmation et la spécification des modèles bigraphiques ; (ii) la visualisation graphique des modèles bigraphiques ; (iii) la vérification des modèles bigraphiques. Sur cette base, un ensemble d'outils est proposé dans la littérature :

BPL Tool [Hojsgaard and Glenstrup, 2011], représente une des premières implémentations des bigraphes. Cet outil permet la manipulation, la simulation et la visualisation des bigraphes. Il peut être utilisé soit à travers une interface utilisateur dans le Web, ou bien comme étant une bibliothèque de programmation.

BigMC [Perrone et al., 2012], est le premier outil de vérification pour les bigraphes, il se base sur la technique de matching (définie dans BPL Tool). Cet outil permet d'explorer les différents états possibles d'un système réactif bigraphique. Une syntaxe appropriée à cet outil a été également proposée afin de spécifier un modèle à base des bigraphes (voir la structure basique dans la figure 3.8) .

```
# Comments
<control definitions>

<names>

<reaction rules>

<model definition>

<properties>

%check;
```

FIGURE 3.8 – Syntaxe du BigMC

Big Red [Faithfull et al., 2012], consiste en un ensemble de plugins Eclipse. Cet outil permet l'édition graphique des bigraphes, avec la possibilité d'exporter les spécifications obtenues en divers formats de fichiers (tel que XML).

DBtk [Bacci et al., 2009], est un outil permettant l'expression des bigraphes via le format standard des graphiques vectoriels ("Scalable Vector Graphics"). Il consiste en un ensemble d'API ("Application Programing Interface"), pour l'édition et la simulation des bigraphes orientés.

BigraphER [Sevegnani and Calder, 2016], acronyme pour "Bigraph Evaluator and Rewriting". Il représente une récente implémentation des bigraphes (avec la possibilité d'éditer les bigraphes partagés). Cet outil comprend un compilateur pour le langage des bigraphes appelé BSL ("BigraphER Specification Language"), et permet la manipulation, la visualisation et la simulation des bigraphes.

Lors de l'exploitation de ces outils, nous avons été confrontés à plusieurs limites. Nous résumons dans ce qui suit les limitations les plus significatives :

— Les outils proposés restent toujours dans un état expérimental, ils n'ont pas connu de suite dans leur développement.
— Leur manipulation est contraignante, ils ne supportent pas tous les concepts des bigraphes pour la représentation textuelle, ainsi que le manque de support pour la génération automatique de leur représentation graphique.
— Problème lors de la mise en œuvre de la dynamique des systèmes réactifs bigraphiques, c'est-à dire, identifier quelle règle de réaction doit-être appliquée pour la réécriture d'un bigraphe donné.
— Bien que BigMC est l'unique outil pour la vérification des bigraphes, il reste très restrictif dans l'expression des propriétés désirées (aucun moyen pour l'expression des propriétés en LTL).

3.8 Conclusion

Dans ce chapitre, nous avons introduit la théorie des bigraphes. Pour ce faire, nous avons d'abord décrit l'anatomie d'un bigraphe ainsi que ces caractéristiques. Nous avons également exposé à travers des exemples les principales opérations algébriques sur les bigraphes. Ensuite, nous avons introduit la dynamique des

systèmes réactifs bigraphiques ainsi que la discipline de typage des bigraphes. Enfin, nous avons décrit quelques extensions et applications de la théorie des bigraphes, et présenté les principaux outils implémentant cette théorie.

La théorie des bigraphes constitue un formalisme mathématique très expressif pour modéliser tant la structure que la dynamique des systèmes Cloud. En effet, le choix des bigraphes est motivé par le fait qu'ils permettent d'exprimer (dans la même structure) deux dimensions différentes : interaction (connectivité) et distribution spatiale (localité). D'autre part, la dynamique d'un système peut être également exprimée au moyen de règles de réaction.

Deuxième partie

Contribution

Chapitre 4

Formalisation d'Architecture Cloud



"That was to me the challenge : picking communicational primitives which could be understood in systems at a reasonably high level as well as in the way these systems are implemented at a very low level...But still, the emphasis ought to be on modelling what happens in real systems, whether they are human-made systems like operating systems, or whether they exist already...But as we move towards mobility, understanding systems that move about globally, you need to commit yourself to a richer set of semantic primitives. I think we are in a terriffic tension between (a) finding a small set of primitives and (b) modelling the real world accurately" –Robin Milner.

Table des matières

4.1 Introduction

Les systèmes cloud connaissent une croissance exponentielle aussi bien en termes de taille que de d'interactions. Pour concevoir ces systèmes de plus en plus complexes, les modèles formels ont gagné en pouvoir d'expression et en abstraction. En effet, en offrant un modèle mathématique d'une architecture cloud, cela permet de gérer cette croissance fulgurante. La théorie des bigraphes [Milner, 2009] offre un modèle mathématique supportant deux dimensions différentes : (1) distribution spatiale des entités physiques ou logicielles, et (2) interaction entre l'ensemble de ces entités. Cette théorie offre également la possibilité de modéliser l'évolution dynamique d'un système à l'aide d'un ensemble de règles de réaction, constituant ainsi les "Systèmes Réactifs Bigraphiques" (BRS). Partant de ce constat, nous les adoptons (pour la première fois [Benzadri et al., 2013a]) en tant que formalisme pour la spécification d'architecture Cloud. Alors, nous associons aux éléments de base du Cloud Computing une sémantique formelle des bigraphes.

Ce chapitre décrit notre contribution à la modélisation d'une architecture Cloud. L'objectif principal est la proposition d'un modèle formel basé bigraphes, prenant en charge les aspects structurels d'une architecture Cloud et constituant une base pour la description de ses reconfigurations. La section 2 sera consacrée à la présentation de l'étude de cas qui sera exploitée pour illustrer les différents concepts introduits dans ce chapitre. La section 3 présente la vue globale de l'architecture Cloud adoptée dans notre approche. Dans la section 4, les différentes couches de l'architecture Cloud adoptée sont formalisées via des bigraphes spécifiques. La section 5 présente une fonction formalisant les quatre modèles de déploiement cloud. Dans la section 6, nous identifions les différentes règles de réaction applicables sur les bigraphes proposés, afin de formaliser les interactions possibles entre les éléments d'un système cloud.

4.2 Exemple de Motivation : "$Cloud\!-\!Healthcare$"

Tout comme les autres secteurs, l'adoption du cloud computing dans le domaine de la santé est de plus en plus émergente car le cloud computing permet de réduire les délais de communication entre d'une part les membres du personnel médical et d'autres part entre les patients et le personnel médical [Grindle et al., 2013]. Il permet également de faciliter l'accès des patients à leurs dossiers médicaux. Cela peut faire face à plus de 60 % des erreurs médicales dues à une mauvaise communication [COCIR, 2012].

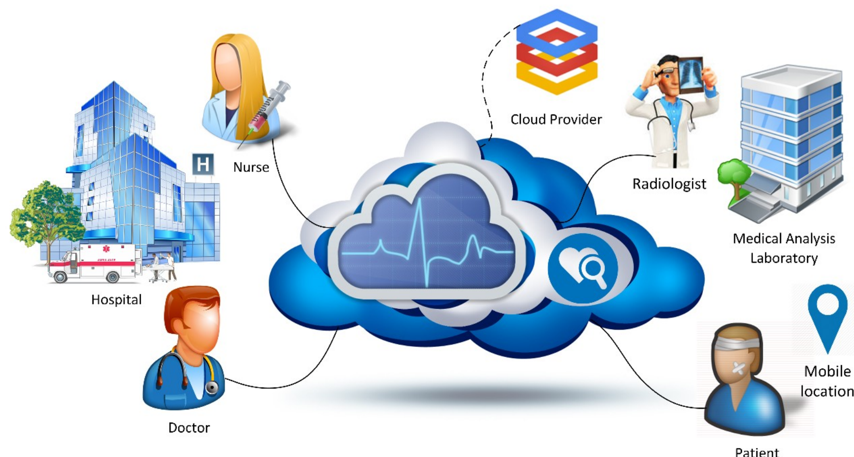


FIGURE 4.1 – Étude de cas : Cloud-Healthcare [Benzadri et al., 2016]

La figure 5.1 met en évidence les différents liens de communications entre le personnel médical, afin d'échanger des informations (données) concernant leurs patients. En conséquence, nous identifions deux organisations (hôpital et laboratoire d'analyses médicales), en plus d'un fournisseur cloud (Google dans notre exemple) possédant le centre de données qui héberge le système "Cloud-Healthcare". En outre, nous suggérons qu'il existe deux acteurs au sein de l'hôpital : un médecin assurant les consultations nécessaires et prescrivant les médicaments essentiels, et une infirmière pour le suivi de l'état des patients. De même, nous identifions un seul acteur dans le laboratoire d'analyses médicales : un radiologue faisant les examens médicaux nécessaires et saisissant les résultats obtenus. Enfin, nous identifions un patient qui peut consulter son PeMR (Patient's Electronic Medical Record) de n'importe quel endroit (un utilisateur mobile).

4.3 Architecture en Couches pour les Systèmes Cloud

L'architecture cloud s'appuie généralement sur des centres de données (cloud datacenters) repartis dans le nuage ; chaque centre de données est composé de milliers de serveurs physiques et virtuels. Ces serveurs sont conçus pour déployer de nombreux services cloud, qui à leur tour répondent aux demandes des utilisateurs dans le cloud. Afin de gérer la complexité de l'architecture cloud, nous avons eu recours à l'un des principaux concepts du génie logiciel, la séparation des préoccupations [De Win et al., 2002]. Ce principe consiste à considérer séparément les différents aspects d'un problème, afin d'en maîtriser la complexité. En effet,

la séparation des préoccupations fournit un support méthodologique permettant de décomposer un système en un ensemble d'éléments compréhensibles, chacun s'intéressant à une seule préoccupation. Ainsi, nous adoptons un style architectural en couches pour modéliser l'architecture cloud. En particulier, nous identifions trois couches différentes : utilisateurs cloud, services cloud et virtualisation cloud (voir figure 4.2). Chaque couche possède ses propres responsabilités et s'appuie sur les autres couches pour assurer ses tâches ;

Couche Utilisateurs Cloud, regroupe l'ensemble des utilisateurs et les interfaces nécessaires pour accéder aux services cloud. Elle concerne l'utilisation directe d'une ressource cloud, telles que : une application par des utilisateurs finaux, une plateforme par des développeurs, ou bien une infrastructure par des administrateurs [Hausman et al., 2013]. De plus, cette couche offre la possibilité d'identifier l'environnement d'accessibilité des utilisateurs dans le cloud ; que ce soit à partir d'un fournisseur cloud, une organisation privée ou bien une position mobile (maison, internet café, aéroport, etc.).

Couche Services Cloud, fournit un niveau supplémentaire d'abstraction en décomposant le système cloud en un ensemble de services spécifiques, pouvant être combinés et réutilisés afin de répondre aux besoins des utilisateurs cloud. Cette couche englobe les modèles les plus répandus des services cloud, à savoir : "Infrastructure as a Service" (IaaS), "Platform as a Service" (PaaS) et "Software as a Service" (SaaS) [Mell and Grance, 2011]. Plus précisément, cette couche décrit les modèles de services cloud ; construits l'un sur l'autre : le niveau PaaS construit sur le niveau IaaS et le niveau SaaS qui tire partie du niveau PaaS [Marks and Lozano, 2010].

Couche Virtualisation Cloud, permet d'assurer une abstraction et isolation des ressources matérielles et logicielles avec une rationalisation de leurs exploitation. En d'autres termes, elle consiste à exécuter plusieurs serveurs virtuels indépendants sur un seul serveur physique puissant [Ou, 2006]. En outre, et en plus de la communication de données (data-communication) entre utilisateurs et services cloud, la couche virtualisation offre la possibilité de transférer une masse de données massives selon trois scénarios de communications : (1) serveur-virtuel vers serveur-virtuel (réseau virtuel), (2) serveur-physique vers serveur-physique (réseau physique) et (3) centre de données vers centre de données (réseau DC).

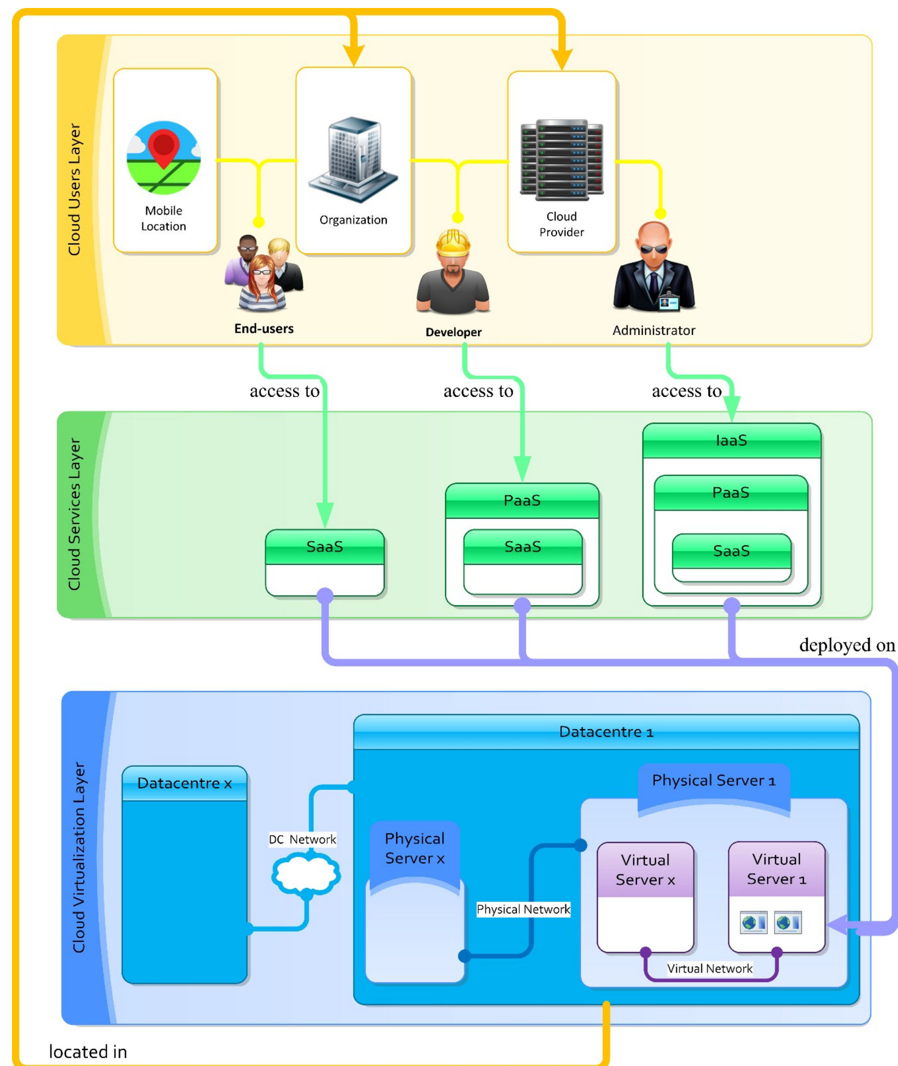


FIGURE 4.2 – Architecture en couches des systèmes cloud [Benzadri et al., 2016]

La figure 4.2 illustre graphiquement ces trois couches ainsi que leurs dépendances. Nous identifions trois principales relations de dépendance entre ces couches :

La relation "Accéder à", entre la Couche Utilisateurs et la Couche Services Cloud. Cette relation aligne les rôles des utilisateurs avec les différents modèles de services cloud ; des administrateurs accédant au IaaS, des développeurs accédant au PaaS et des utilisateurs finaux accédant au SaaS seulement.

La relation "Déployer sur", entre la Couche Services Cloud et la Couche Virtualisation Cloud, pour indiquer que les services cloud peuvent être exécutés sur les serveurs virtuels correspondants.

La relation "Situer dans", entre la Couche Virtualisation Cloud et la couche Utilisateurs Cloud, pour indiquer que les centres de données dans le nuage pouvant être hébergés dans les locaux de l'organisation (cloud privé) ou bien dans les locaux d'un fournisseur cloud (cloud publique).

4.4 Modélisation d'une Architecture Cloud

Dans le domaine du génie logiciel, les méthodes formelles sont une collection de notations et techniques, visant à améliorer la qualité de conception des systèmes. Cependant, les modèles formels doivent non seulement aider à obtenir une meilleure représentation, mais aussi une vision plus abstraite (et modulaire) de l'architecture des systèmes cloud. Un modèle formel caractérisant l'architecture cloud présentée dans la section précédente, est défini par un bigraphe spécifique, appelé : CAB "Cloud Architecture Bigraph". Ce bigraphe est la composition de trois bigraphes élémentaires ; le premier "Cloud Users Bigraph" (CUB) formalisant la couche utilisateurs cloud, le second "Cloud Services Bigraph" (CSB) formalisant la couche services cloud, et le troisième "Cloud Virtualization Bigraph" (CVB) formalisant la couche virtualisation cloud.
Nous détaillerons dans ce qui suit la définition de chaque bigraphe du modèle ainsi obtenu.

4.4.1 Formalisation de la Couche Utilisateurs

Nous proposons la définition formelle (voir la définition 4.1) du CUB (Cloud Users Bigraph) qui capture les différents concepts liés à la couche utilisateurs cloud (voir figure 4.2). Ainsi, nous modélisons les utilisateurs par des nœuds avec trois contrôles (types) spécifiques : EU (Utilisateur final), DEV (Développeur) et AD (Administrateur). Des ports sont prévus dans chaque nœud utilisateur afin de lui permettre d'émettre des demandes de service à travers internet. Nous identifions également trois autres contrôles (types de nœuds) représentant : une organisation (ORG), un fournisseur cloud (CCP) et une situation mobile (MLC). De ce fait, nous utilisons la notion de région (root) de la théorie des bigraphes pour représenter la localisation des utilisateurs dans le cloud. Un site représentant un espace d'hébergement pour les centres de données, est prévu dans chaque nœud de type organisation (ORG) ou fournisseur cloud (CCP). Enfin, l'interface externe du CUB représente les interfaces de communication possibles entre un utilisateur et un service dans le Cloud. Le tableau 4.1 récapitule les différents

concepts liés à la couche utilisateurs cloud avec une sémantique basée bigraphe.

TABLE 4.1 – Sémantique liée aux Utilisateurs Cloud

Cloud Computing	**Bigraphe (CUB)**	**Représentation graphique**
Utilisateur final (EU), Développeur (DEV) et Administrateur (AD)	Nœud Atomique	
Organisation (ORG), Fournisseur cloud (CCP) et Situation mobile (MLC)	Nœud Actif	
Accès Internet (IAC)	Port	
Espace d'hébergement pour centre de données (DcHS)	Site	
Localisation des utilisateurs (LC)	Région (Root)	
Interface de communication d'utilisateurs (UCI)	Interface externe (Y_{CUB})	

Définition 4.1 ***"Cloud Users Bigraph"***
Le bigraphe CUB ("Cloud Users Bigraph") formalisant la couche Utilisateurs Cloud prend la forme :
$(V_{CUB}, E_{CUB}, ctrl_{CUB}, GP_{CUB}, GL_{CUB}) :< m_{CUB}, X_{CUB} > \rightarrow < n_{CUB}, Y_{CUB} >$, *dont :*

— V_{CUB} *est l'ensemble des nœuds, représentant des utilisateurs cloud ;*
— E_{CUB} *est un ensemble fini d'arêtes, reliant les demandes des utilisateurs ;*
— $ctrl_{CUB} : V_{CUB} \rightarrow S_{CUB}$ *est une transformation attribuant un contrôle à chaque nœud, dont la signature* S_{CUB} *= EU : (x, atomic), DEV : (x, atomic), AD : (x, atomic), ORG : (0, actif), CCP : (0, actif), MLC : (0,actif) ;*
— GP_{CUB} *est le graphe de places spécifiant les localités des utilisateurs cloud, et* GL_{CUB} *est le graphe de liens exprimant la connectivité des utilisateurs cloud ;*
— m_{CUB} *est le nombre de sites représentant des espaces d'hébergement pour les centres de données, et* n_{CUB} *est le nombre de régions décrivant les localisations des utilisateurs ;*

— *X_{CUB} est l'interface interne, et Y_{CUB} est l'interface externe spécifiant des interfaces de communication pour les utilisateurs cloud.*

Exemple

En se basant sur cette formalisation, la figure 4.3 représente le CUB associé à l'étude de cas présentée dans la section 2 de ce chapitre. Les deux organisations identifiées (le laboratoire d'analyses médicales et l'hôpital), sont respectivement représentées via les nœuds : $O1, O2 \in V_{CUB}$, dont : $ctrl(O1) = ctrl(O2) = "ORG"$, par contre le fournisseur cloud est représenté via le nœud : $O3 \in V_{CUB}$, dont : $ctrl(O3) = "CCP"$. Nous suggérons que le laboratoire d'analyses médicales (O1) et le fournisseur cloud (O3), contiennent tous les deux un espace d'hébergement pour centre de données, spécifié par les sites (S0 et S1). En outre, le radiologue, le médecin, l'infirmière et le patient sont spécifiés via les nœuds : $C1, C2, C3, C4 \in V_{CUB}$, dont les contrôles sont respectivement : $ctrl(C1) = ctrl(C2) = ctrl(C3) = ctrl(C4) = "EU"$. Alors que le radiologue, le médecin et l'infirmière sont au sein de leurs organisations, le patient accède aux services cloud à partir d'une situation mobile (codée par le nœud $n1 \in V_{CUB}$, où $ctrl(n1) = "MLC"$). Chacun des nœuds utilisateurs (C1, C2, C3 et C4) possède des ports représentant un accès à Internet pour envoyer leurs demandes de service dans le cloud. Ces demandes de service sont formellement regroupées dans les noms externes ($Y1, Y2, Y3, Y4 \in Y_{CUB}$) du bigraphe proposé.

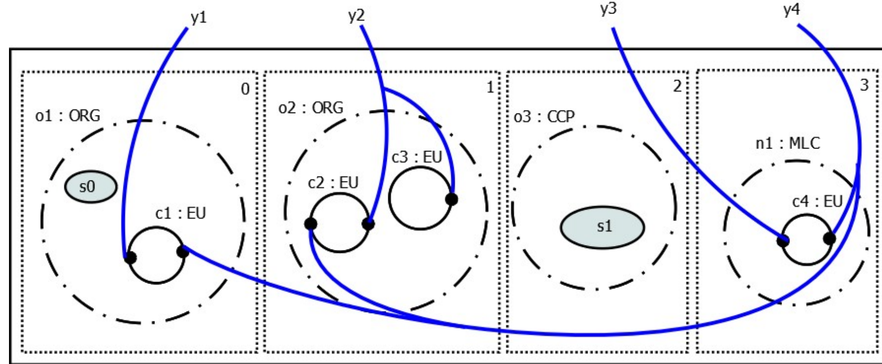


FIGURE 4.3 – Exemple d'application du bigraphe CUB

Ainsi, le bigraphe des utilisateurs cloud associé à l'étude de cas "CloudHealthcare" (voir la figure 4.3) est défini par $< 2, \phi > \rightarrow < 4, \{Y1, Y2, Y3, Y4\} >$. Il est constitué de deux sites, quatre racines et un ensemble de noms externes $\{Y1, Y2, Y3, Y4\}$.

4.4.2 Formalisation de la Couche Services

Nous proposons la définition formelle (voir définition 4.2) du bigraphe CSB "Cloud Services Bigraph" qui supporte les concepts liés à la couche services cloud. Les services cloud disponibles sont modélisés par des nœuds dans le CSB. Les contrôles attachés à ces nœuds permettent de distinguer entre les trois modèles de services (SaaS, PaaS, et IaaS). Chaque nœud service possède un port qui le relie directement avec d'éventuels utilisateurs possibles. Les clusters de services cloud sont spécifiés via la notion de région ("Root") dans les bigraphes, tandis que leurs interfaces de communication sont représentées par les interfaces internes du CSB. En outre, la hiérarchie des services cloud est formellement spécifiée dans la définition 4.2 à travers des contraintes associées aux types de noeuds. Nous définissons trois contraintes entre les différents nœuds de services cloud : (1) un nœud de contrôle IaaS ne peut contenir que des nœuds de contrôle PaaS, (2) un nœud de contrôle PaaS ne peut contenir que des nœuds de contrôle SaaS, et enfin, (3) un nœud de contrôle SaaS ne peut contenir aucun autre nœud. Sur cette base, nous suggérons que les nœuds de contrôles IaaS et PaaS soient actifs, alors que les nœuds de contrôle SaaS soient atomiques. Le tableau 4.2 résume les principaux concepts liés aux services cloud avec leur sémantique bigraphique associée.

TABLE 4.2 – Sémantique liée aux Services Cloud

Cloud Computing	**Bigraphe (CSB)**	**Représentation graphique**
Service de type "SaaS"	Nœud Atomique	
Service de type "PaaS", et Service de type "IaaS"	Nœud Actif	
Cluster de services (SvC)	Région (Root)	
Accès service-utilisateur (SCA)	Port	
Interface de communication des services (SCI)	Interface interne (X_{CSB})	

Définition 4.2 ***"Cloud Services Bigraph"***
Le bigraphe CSB formalisant les services cloud prend la forme :
$(V_{CSB}, E_{CSB}, ctrl_{CSB}, GP_{CSB}, GL_{CSB}) :< m_{CSB}, X_{CSB} > \rightarrow < n_{CSB}, Y_{CSB} >$. *Où :*

— *V_{CSB} est un ensemble de nœuds représentant les services cloud, Chaque nœud service doit satisfaire une des trois contraintes :*

1. $\forall U, N \in V_{CSB}, (U.N \wedge ctrl_{CSB}(U) = IaaS) \Longrightarrow ctrl_{CSB}(N) = PaaS,$
2. $\forall U, N \in V_{CSB}, (U.N \wedge ctrl_{CSB}(U) = PaaS) \Longrightarrow ctrl_{CSB}(N) = SaaS,$
3. $\forall U \in V_{CSB}, ctrl_{CSB}(U) = SaaS \Longrightarrow v \in V_{CSB} | U.N = \phi.$ *;*

— *E_{CSB} est un ensemble fini d'arêtes reliant les offres de services cloud ;*

— *$ctrl_{CSB} : V_{CSB} \rightarrow S_{CSB}$ attribue un contrôle (type) à chaque nœud service, dont la signature S_{CSB}= IaaS : (1, actif), PaaS : (1, actif), SaaS : (1, atomic) ;*

— *GP_{CSB} représente le graphe de places et GL_{CSB} représente le graphe de liens ;*

— *m_{CSB} est le nombre de sites et n_{CSB} est le nombre régions exprimant les clusters de service cloud ;*

— *X_{CSB} est l'interface interne représentant des interfaces de communication des services cloud, et Y_{CSB} est la interface externe.*

Exemple

La couche service du système "Cloud-Healthcare" doit satisfaire les demandes du personnel médical, en leur fournissant les services nécessaires pour l'échange d'information relative à l'état d'un patient. La figure 4.4 représente la formalisation de notre cas à travers le CSB. Nous identifions ainsi les services cloud suivants :

— $i1 \in V_{CSB}$ / $ctrl(i1)$ = "$IaaS$" et $pt1 \in V_{CSB}$ / $ctrl(pt1)$ = "$PaaS$", représentant respectivement deux services cloud (infrastructure et plateforme) pour l'accueil des services liés au laboratoire d'analyses médicales (S1 et S4) ;

— $i2 \in V_{CSB}$ / $ctrl(i2)$ = "$IaaS$",et $pt2 \in V_{CSB}$ /$ctrl(pt2)$ = "$PaaS$", représentant respectivement deux services cloud (infrastructure et plateforme), pour l'hébergement des services liés à l'hôpital (S2 and S3) ;

— $S4, S1 \in V_{CSB}$ / $ctrl(S4) = ctrl(S1)$ = "$SaaS$", permettent respectivement de consulter et d'éditer les résultats d'examens médicaux d'un patient ;

— $S2, S3 \in V_{CSB}$ / $ctrl(S2) = ctrl(S3)$ = "$SaaS$", permettent respectivement de consulter et d'éditer le PeMR ("Patient electronic Medical Records") ;

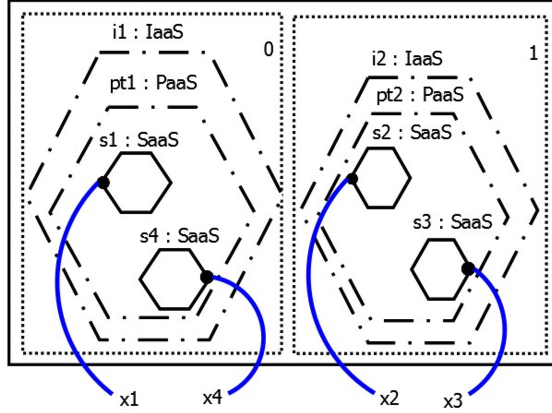


FIGURE 4.4 – Exemple d'application du bigraphe CSB

Les différents services fournis par le système "Cloud-Healthcare" sont accessibles via les noms internes ($X1, X2, X3, X4 \in X_{CSB}$) du CSB. Ainsi, le bigraphe $< 0, \{x1, x2, x3, x4\} > \rightarrow < 2, \phi >$ spécifie algébriquement la couche service de notre cas d'étude. Il n'a pas de sites, possède deux racines et un ensemble de noms internes $\{X1, X2, X3, X4\}$.

4.4.3 Formalisation de la Couche Virtualisation

La définition 4.3 (du "Cloud Virtualization Bigraph", CVB) intercepte et formalise les différents concepts liés à la couche virtualisation cloud. Un centre de données se compose de plusieurs serveurs physiques exécutant des serveurs virtuels individuels. Ainsi, nous définissons les centres de données, les serveurs physiques et virtuels par des nœuds avec trois contrôles spécifiques : DC (Centre de données), VS (Serveur Virtuel) et PS (Serveur Physique). Les nœuds de contrôle (DC) peuvent contenir plusieurs nœuds de contrôle (PS), qui à leur tour contiennent d'autres nœuds de contrôle (VS). De plus, un site est fourni dans chaque nœud de contrôle (VS) ; représentant un espace d'hébergement pour le déploiement de nouveaux services cloud sur les serveurs virtuels. Par conséquent, nous suggérons que les trois contrôles des nœuds identifiés (DC, PS et VS) soient actifs. Un port est fourni dans chaque nœud du CVB, spécifiant les interfaces de communication entre centres de données, serveurs physiques et serveurs virtuels. Ainsi, le graphe de liens du CVB, permet la représentation des différents scénarios de communication disponibles dans le cloud [MRV-Communications, 2013]. Enfin, la notion de région (root) dans le CVB permet de délimiter la notion de clusters de serveurs.

Le tableau 4.3 recense les différents concepts liés à la couche Virtualisation Cloud, ayant chacun une sémantique précise à base des bigraphes.

TABLE 4.3 – Sémantique liée à la Virtualisation Cloud

Cloud Computing	**Bigraphe(CVB)**	**Représentation graphique**
Centre de données (DC)	Nœud Actif	
Serveur Virtuel (VS)	Nœud Actif	
Serveur Physique (PS)	Nœud Actif	
Cluster de serveur (SrC)	Région (Root)	
Espace d'hébergement de services (SvHS)	Site	
Interfaces de communication (DC-DC, PS-PS et VS-VS)	Ports	

Définition 4.3 ***"Cloud Virtualization Bigraph"***
Le bigraphe CVB formalisant la couche virtualisation cloud prend la forme : $(V_{CVB}, E_{CVB}, ctrl_{CVB}, GP_{CVB}, GL_{CVB}) :< m_{CVB}, X_{CVB} >\to< n_{CVB}, Y_{CVB} >$. *Où :*

— V_{CVB} *est l'ensemble des nœuds représentant les centres de données et les serveurs physiques et virtuels ;*
— E_{CVB} *est un ensemble fini d'arêtes connectant les centres de données, les serveurs physiques et les serveurs virtuels ;*
— $ctrl_{CVB} : V_{CVB} \to S_{CVB}$ *une transformation qui associe un contrôle à chaque nœud serveur où la signature* S_{CVB} *est définie par* S_{CVB} *= PS : (1, Actif), VS : (1, Actif), DC : (1, Actif) ;*
— m_{CVB} *est le nombre de sites décrivant les espaces d'hébergement des services cloud et* n_{CVB} *est le nombre de régions exprimant les clusters de serveurs ;*
— X_{CVB} *est l'interface interne et* Y_{CVB} *est l'interface externe ;*
— GP_{CVB} *représente le graphe de places et* GL_{CVB} *représente le graphe de liens.*

Exemple

Nous supposons que le "Cloud-Healthcare" fonctionne sur deux centres de données différents, le premier contient un serveur physique hébergeant un seul serveur virtuel, tandis que le deuxième centre de données contient un serveur physique hébergeant deux serveurs virtuels. La figure 4.5 représente la formalisation

de ce cas à travers le CVB. Nous modélisons les centres de données, les serveurs physiques et virtuels via les nœuds suivants : $dc1, dc2, p1, p2, v1, v2, v3 \in V_{CVB}$ dont les contrôles sont respectivement : $ctrl(dc1) = ctrl(dc2) = "DC", ctrl(p1) = ctrl(p2) = "PS", ctrl(v1) = ctrl(v2) = ctrl(v3) = "VS"$. Les sites S0 et S1 disponibles dans les serveurs virtuels "v1 et v2" servent à héberger de futurs services cloud.

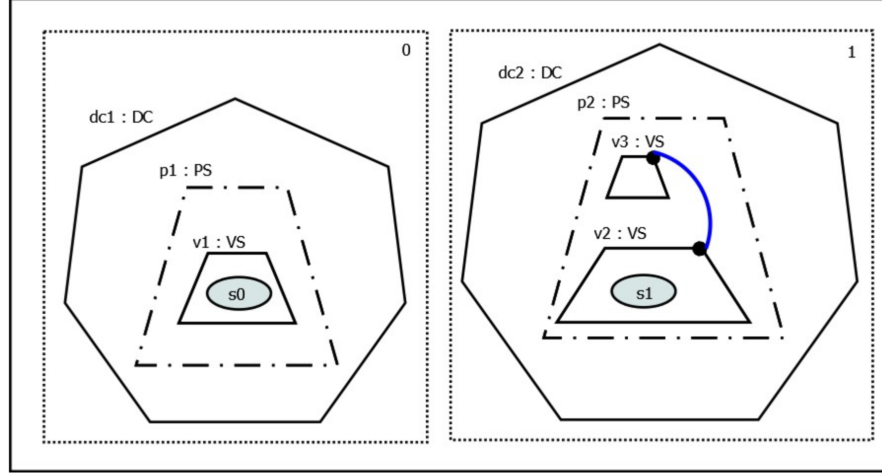


FIGURE 4.5 – Exemple d'application du bigraphe CVB

Enfin, nous écrivons $< 2, \phi > \rightarrow < 2, \phi >$ pour décrire le bigraphe formalisant notre cas. Il possède seulement deux sites et deux régions.

4.4.4 Définition du Modèle Composé "CAB"

L'adoption d'un style architectural en couches pour les systèmes cloud, permet au concepteur de gérer la complexité de ces systèmes avec la possibilité de spécifier formellement chaque couche via un bigraphe dédié. Par ailleurs, l'opération de composition des bigraphes permet de construire des bigraphes plus grands pour une vue globale du système à spécifier. Le bigraphe CAB ("Cloud Architecture Bigraph"), résultant de la composition des trois bigraphes définis (CUB, CSB et CVB), formalise les différentes relations entre les trois couches identifiées (voir figure 4.2) :

1. La composition des bigraphes CUB et CSB, modélise les communications possibles entre les utilisateurs et les services cloud, elle exprime des liens de demande de service. Cela est formalisé en liant les noms externes du CUB avec les noms internes correspondants du CSB.
2. La composition des bigraphes CSB et CVB, se réfère à la localisation des services cloud au sein des serveurs virtuels. Dans ce cas, les racines du

CSB sont fusionnées avec les sites du CVB. Elle exprime le déploiement de services cloud dans des machines virtuelles.

3. La composition des bigraphes CVB et CUB, modélise la localisation des centres de données vis-à-vis des utilisateurs, les organisations et les fournisseurs cloud. Cela est formalisé via le remplacement des sites du CUB par les racines du CVB.

Le "Cloud Architecture Bigraph", formalisant tous les éléments architecturaux d'un système cloud, se compose de deux structures indépendantes : un graphe de places spécifiant la distribution spatiale des entités architecturales et un graphe de liens définissant leurs interactions. Sur cette base, nous définissons séparément dans ce qui suit (voir définition 4.4) la composition des graphes de places et graphes de liens (des trois bigraphes : CUB, CSB et CVB), nous les regroupons ensuite pour définir le CAB. A noter que l'ordre des sites et racines dans la composition des trois bigraphes (CUB, CSB et CVB) à de l'importance. De plus, le nombre de racines dans CSB doit être égal au nombre de sites dans CVB. De même le nombre de racines dans CVB doit être égal au nombre de sites dans CUB.

Définition 4.4 ***"Cloud Architecture Bigraph"***
Le bigraphe CAB est obtenu en composant les trois bigraphes suivants : CUB, CVB et CSB ;

$$CAB = CUB \circ (CVB \circ CSB) =< CAB^P, CAB^L >: I_{CAB} \to J_{CAB}$$

— *A partir des graphes de places composites* $CSB^P : a \to b, CVB^P : b \to c, CUB^P : c \to d$, *nous formons le graphe de places :* $CAB^P = (V_{CAB}, ctrl_{CAB}, prnt_{CAB}) : a \to d$, *tel que :* $V_{CAB} = V_{CSB} \uplus V_{CVB} \uplus V_{CUB}$, *la transformation de contrôle* $ctrl_{CAB} = ctrl_{CSB} \uplus ctrl_{CVB} \uplus ctrl_{CUB}$ *et la transformation de parenté (indiquant le parent de chaque nœud) est définie comme suit : si* $w \in a \uplus V_{CAB}$ *est un site ou un nœud de CAB alors* $prnt_{CAB}(w) =_{def}$
  — $prnt_{CVB}(w)$ *si* $w \in b \uplus V_{CVB}$ *et* $prnt_{CVB}(w) \in V_{CVB}$ *;*
  — $prnt_{CSB}(w)$ *si* $w \in a \uplus V_{CSB}$ *et* $prnt_{CSB}(w) \in V_{CSB}$ *;*
  — $prnt_{CUB}(j)$ *si* $w \in b \uplus V_{CVB}$ *et* $prnt_{CVB}(w) = j \in c$ *;*
  — $prnt_{CVB}(j)$ *si* $w \in a \uplus V_{CSB}$ *et* $prnt_{CSB}(w) = j \in b$ *;*
  — $prnt_{CUB}(w)$ *si* $w \in V_{CUB}$ *.*

— *A partir des graphes de liens composites* $CSB^L : X \to Y, CVB^L : Y \to Z, CUB^L : Z \to W$, *nous formons le graphe de liens :* $CAB^L = (V_{CAB}, E_{CAB}, ctrl_{CAB}, link_{CAB}) : X \to W$, *tel que :* $V_{CAB} = V_{CSB} \uplus V_{CVB} \uplus V_{CUB}$, *la transformation de contrôle* $ctrl_{CAB} = ctrl_{CSB} \uplus ctrl_{CVB} \uplus ctrl_{CUB}$

, l'ensemble des arcs : $E_{CAB} = E_{CSB} \uplus E_{CVB} \uplus E_{CUB} \uplus Z$ *et la transformation de liens (reliant les ports aux arcs) est définie comme suit : si* $p \in X \uplus P_{CSB} \uplus P_{CVB} \uplus P_{CUB}$ *est un port a relier avec un arc dans* E_{CAB} *alors* $link_{CAB}(p) = e$ *dont :*
— $e \in E_{CSB}$*, ou* $e \in E_{CUB}$*, si* $p \in X \uplus P_{CSB}$ *;*
— $e \in E_{CVB}$*, si* $p \in P_{CVB}$ *;*
— $e \in E_{CUB} \uplus W$*, si* $p \in P_{CUB}$*.*

Exemple

Le système "Cloud-Healthcare" spécifié via le bigraphe CAB, est représenté dans la figure 4.6. En effet, une seule structure prend en compte tous les éléments architecturaux au sein des trois couches identifiées (utilisateurs cloud, services cloud et virtualisation cloud). En se référant à notre définition 4.4, le bigraphe donné par $< 0, \phi > \rightarrow < 4, \phi >$ est la composition des trois bigraphes définis dans les exemples précédents :

- Le bigraphe des services Cloud (CSB), noté dans notre cas par $< 0, x1, x2, x3, x4 > \rightarrow < 2, \phi >$;
- Le bigraphe de virtualisation Cloud (CVB), donné dans notre cas par $< 2, \phi > \rightarrow < 2, \phi >$;
- Le bigraphe des utilisateurs Cloud (CUB), décrit dans notre cas par $< 2, \phi > \rightarrow < 4, Y1, Y2, Y3, Y4 >$.

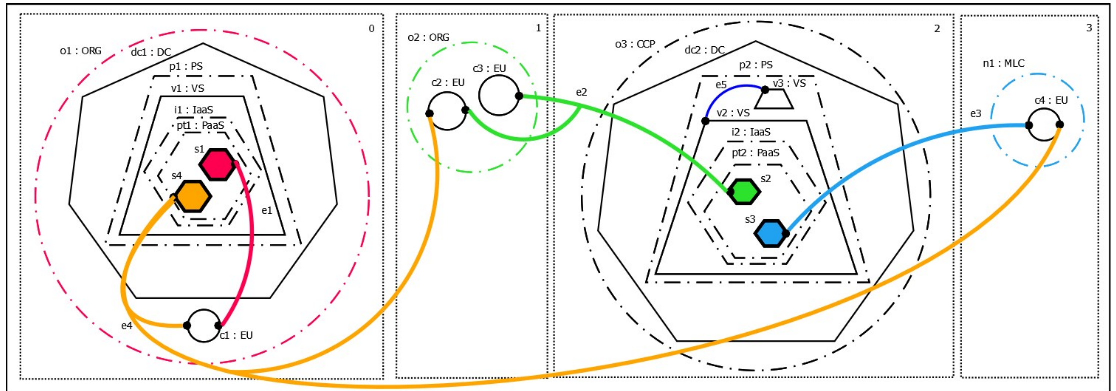


FIGURE 4.6 – Exemple d'application du CAB

Selon la figure 4.6, les centres de données (dc1 et dc2 du CVB) ont été placés respectivement dans leurs espaces d'hébergement au sein du "O1" (le laboratoire

d'analyses médicales) et "O2" (le fournisseur cloud) du bigraphe CUB. Ensuite, les services cloud : S1, S2, S3 et S4 du CSB, ont été respectivement déployé sur les serveurs virtuels v1 et v2 du CVB. Enfin, l'allocation des services cloud (S1, S2, S3 et S4 du CSB) par les utilisateurs (C1, C2, C3 et C4 du CUB) est réalisée via l'ensemble des arêtes : e1, e2, e3 et e4.

En plus de la spécification des éléments architecturaux des trois couches cloud, le bigraphe formalisant notre étude de cas via le CAB, prend en charge à la fois l'interaction entre les entités du système "Cloud-Healthcare" ainsi que leur distribution spatiale. Cela est effectué via les deux structures indépendantes du bigraphe, à savoir :

— Le graphe de places, spécifiant la hiérarchie des entités du système "Cloud-Healthcare" (comme illustré dans la figure 4.7) ;
— Le graphe de liens ; spécifiant les connectivités possibles entre ces éléments (comme illustré dans la figure 4.8).

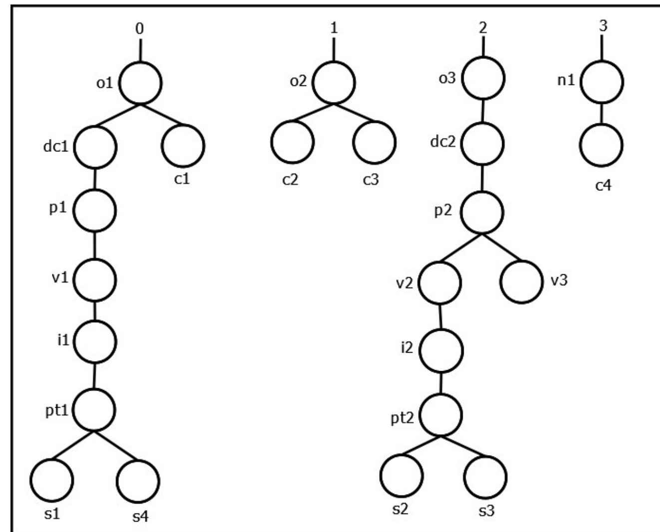


FIGURE 4.7 – Exemple d'application du CAB –graphe de places

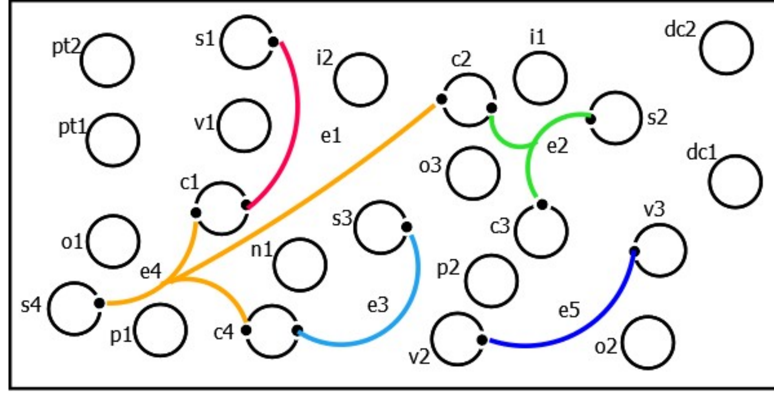


FIGURE 4.8 – Exemple d'application du CAB –graphe de liens

4.5 Formalisation des Modèles de Déploiement Cloud

En se basant sur la définition NIST du cloud computing [Mell and Grance, 2011], les quatre modèles de déploiement des services cloud peuvent être identifiés en fonction de leurs emplacement au sein des centres de données vis-à-vis des utilisateurs dans le cloud ; un centre de données appartenant à un fournisseur cloud (externe) ou bien un centre de données que l'organisation (de l'utilisateur) possède (interne). Nous proposons (voir la définition 4.5) une fonction qui associe à chaque service cloud son modèle de déploiement *(public, privé, communautaire, hybride).*

Définition 4.5 ***Modèles de Déploiement Cloud***
Étant donné un bigraphe $CAB = \{V_{CSB} \uplus V_{CVB} \uplus V_{CUB}, E_{CAB}, ctrl_{CSB} \uplus ctrl_{CVB} \uplus ctrl_{CUB}, link_{CAB}\}$*, soit* $s \in V_{CSB}$ *un service cloud,* $dc \in V_{CVB} | prnt^3(s) = dc$ *un centre de données hébergeant le service cloud s,* $u \in V_{CUB}$ *un utilisateur cloud,* $ps \in P$ *un port du nœud service,* $pu \in P$ *un port du nœud utilisateur et* $es \in E_{CAB}$ *une arête reliant le service "s" à son utilisateur "u".*
Nous définissons la fonction :

$$getCloudT : V_{CSB} \rightarrow \{ \text{ Public, Privé, Communautaire, Hybride } \}$$

qui associe à chaque service cloud ($s \in V_{CSB}$*) son modèle de déploiement qui peut être soit Public, Privé, Communautaire ou Hybride ; de la manière suivante :*

— *si* $ctrl(prnt(u)) = "MLC"$ *et* $ctrl(prnt(dc)) = "CCP" \rightarrow getCloudT(s) =$ *Public ;*

— *si* $(prnt(u) = prnt(dc))$ *et* $(ctrl(prnt(dc)) = "ORG") \rightarrow getCloudT(s) =$ *Privé ;*

— *si* $(\forall u' \in V_{CUB} | link_{CAB}(u', pu) = es)$ *et* $(prnt(u) = prnt(u'))$ *et* $(ctrl(prnt(u)) = "ORG") \rightarrow getCloudT(s) = Communautaire$;
— *si* $(\exists u' \in V_{CUB} | link_{CAB}(u', pu) = es)$ *et* $(prnt(u)! = prnt(u'))$ *et* $(ctrl(prnt(dc) = "ORG")) \rightarrow getCloudT(s) = Hybride$.

Une conséquence intéressante qui découle de la définition 4.4 du bigraphe composé CAB, est cette fonction proposée définissant les quatre types de déploiement : cloud public –appartenant à un fournisseur (ctrl (prnt (dc)) = "CCP") et mis à la disposition du grand public (ctrl (prnt (u)) = "MLC"), cloud privé –détenu par une seule organisation (ctrl (prnt (dc) = "ORG")) et mis à disposition uniquement à ses utilisateurs (prnt (u) = prnt (dc)), cloud communautaire –limité à certains utilisateurs qui ont des intérêts communs (prnt (u) = prnt (u ')), et enfin, le cloud hybride –combinant deux ou plusieurs cloud distincts (public, privé ou communautaire).

Exemple

En se référant à l'étude de cas du système "Cloud-Healthcare" (voir la figure 4.6), la fonction proposée (dans la définition 4.5) retourne pour notre cas les valeurs suivantes :

— Le service cloud (s3) est détenu par le fournisseur cloud (O3, dont : ctrl (O3) = ctrl (prnt (dc2)) = "CCP") et mis à la disposition des patients (c4, dont : $link_{CAB}(c4, p2) = e3$ et $link_{CAB}(s3, p) = e3$) de n'importe où (ctrl (prnt (c4)) = ctrl (n1) = "MLC"). Ainsi, le service cloud (s3) est déployé en tant que cloud public (getCloudT (s3) = Public).
— Le service cloud (s2) est limité aux médecins (c2) et infirmières (c3) qui ont des intérêts communs (prnt (c2) = prnt (c3)). Ainsi, le service cloud (s2) est déployé en tant que cloud communautaire (getCloudT (s2) = Communauté).
— Le service cloud (s1) est détenu par le laboratoire d'analyses médicales (ctrl(o1) = ctrl(prnt(dc1)) = "ORG") et mis à la disposition uniquement des radiologues (c1, où : prnt (c1) = prnt (dc1)). Ainsi le service cloud (s1) est déployé en tant que cloud privé (getCloudT (s1) = Privé).
— Le service cloud (s4) est détenu par le laboratoire d'analyses médicales (ctrl (o1) = ctrl (prnt (dc1)) = "ORG") et mis à la disposition des radiologues, médecins et patients (c1, c2, c4, où : Prnt (c1) ! = prnt (c2) != prnt (c4)). Ainsi, le service cloud (s4) est déployé en tant que cloud hybride (getCloudT (s4) = Hybride).

4.6 Modélisation de la Dynamique d'un Système Cloud

L'architecture cloud peut subir différentes reconfigurations dynamiques afin d'optimiser les ressources cloud et d'accroître leur disponibilité. Ces reconfigurations permettent de modéliser le comportement d'un système cloud face aux événements internes ou externes auxquels il peut être exposé. Étant donné qu'un BRS (système réactif bigraphique) est constitué d'un ensemble des règles de réaction définissant la dynamique des bigraphes, nous utilisons cette notion pour parvenir à la formalisation des réactions devant être prises lors du déclenchement d'un événement du système cloud.

Dans cette section, nous introduisons un ensemble de règles de réaction bigraphiques afin de décrire l'évolution des systèmes cloud. En particulier, les règles proposées formalisent les reconfigurations architecturales dans chacune des trois couches identifiées (utilisateurs cloud, services cloud, et virtualisation cloud). En d'autres termes, les règles proposées permettent de formaliser les différents scénarios de fédération dans le cloud [Kurze et al., 2011] : Migration et Réplication de ressources cloud. Le tableau (4.4) résume l'ensemble de règles de réaction proposées (CUR –"Cloud Users Rules", CSR –"Cloud Services Rules" et CVR –"Cloud Virtualization Rules").

TABLE 4.4 – Règles de réaction proposées

ID règle	Bigraphe concerné	Description
R1	Cloud Users Bigraph	Changement de localisation des utilisateurs.
R2		Connexion d'un nouvel utilisateur.
R3		Déconnexion d'un utilisateur.
R4	Cloud Services Bigraph	Migration d'un service cloud.
R5		Réplication d'un service cloud.
R6		Déploiement d'un nouveau service cloud.
R7		Suppression d'une instance de service cloud.
R8	Cloud Virtualization Bigraph	Migration d'un serveur virtuel.
R9		Réplication d'un serveur virtuel.
R10		Déploiement d'un nouveau serveur virtuel.
R11		Suppression d'une instance de serveur virtuel.
R12		Consolidation d'un serveur physique.

Reconfiguration de la Couche Utilisateurs

Dans le but de modéliser le comportement de la couche utilisateurs spécifiée par le bigraphe "Cloud Users Bigraph", nous jugeons utile d'identifier trois règles dont chacune assure une fonctionnalité particulière ;

La règle R1 modélise le changement d'emplacement d'un utilisateur cloud, que ce soit d'une organisation à une autre, ou vers une autre position (mobile). Sa forme algébrique est définie par :

$$R1 :: o1.c1_y||o2 \rightarrow o1||o2.c1_y$$

Dans les deux côtés de la règle, le nœud utilisateur cloud (c1) peut être de type : utilisateur final (ctrl (c1) = "EU"), développeur (ctrl (c1) ="DEV") ou administrateur (ctrl (c1) = "AD"), tandis que les nœuds parents (o1 et o2) peuvent être de sorte : organisation (ctrl (o1) = ctrl (o2) = "ORG") ou situation mobile (ctrl (o1) = ctrl (o2) = "MLC").

La règle R2 spécifie la connexion d'un nouvel utilisateur au système cloud. Ceci est défini via l'inclusion d'un nouveau nœud reliant un nom externe (y) du CUB. La forme algébrique de cette règle est présentée comme suit :

$$R2 :: o1 \rightarrow o1.c1_y$$

Dans le coté gauche de la règle, le nœud (o1) peut être de contrôle : organisation (ORG) ou situation mobile (MLC), tandis que sur le coté droit, le nœud utilisateur peut être de contrôle : utilisateur final (ctrl (c1) = "EU"), développeur (ctrl (c1) = "DEV") ou administrateur (ctrl (c1)= "AD").

La règle R3 de cette catégorie permet de déconnecter un utilisateur présent dans le système cloud. Elle est formalisée dans les bigraphes par la suppression de tous les liens auxquels participe le nœud utilisateur, ensuite la suppression physique de ce nœud. Dans ce qui suit, nous illustrons la forme algébrique de cette règle :

$$R3 :: o1.c1_y \rightarrow o1$$

Cette règle peut être appliquée aux différents contrôles du nœud utilisateur (c1) ; que ce soit : utilisateur final (ctrl (c1) = "EU"), développeur (ctrl (c1) = "DEV") ou administrateur (ctrl (c1)= "AD").

Reconfiguration de la Couche Services

Toutes les reconfigurations possibles sur le bigraphe "Cloud Services Bigraph", sont formalisées par les règles de rection qui lui sont associées. Ainsi, nous proposons quatre règles décrivant la dynamique de ces services ;

La règle R4 modélise la migration d'un service cloud d'un fournisseur cloud vers un autre. Sa forme algébrique est définie comme suit :

$$R4 :: o.(dc.(p.(v.(i.s_x))))||o'.(dc'.(p'.(v'))) \rightarrow$$
$$o.(dc.(p.(v)))||o'.(dc'.(p'.(v'.(i.s_x))))$$

Dans les deux côtés de cette règle, le nœud service (s) peut être de contrôle : "PaaS" ou "SaaS", tandis que les nœuds (o et o') sont de contrôle : fournisseur cloud "CCP" ou organisation "ORG".

La règle R5 permet la réplication ou la création d'une copie d'un service dans différents fournisseurs cloud, suite à une montée en charge des requêtes utilisateurs. La réplication concerne les trois types de services cloud, à savoir IaaS, PaaS et SaaS. Dans ce qui suit, nous présentons sa forme algébrique :

$$R5 :: o.(dc.(p.(v.(i.s_x))))||o'.(dc'.(p'.(v'))) \rightarrow$$
$$o.(dc.(p.(v.(i.(s_x)))))||o'.(dc'.(p'.(v'.(s'_x))))$$

dans le côté gauche de la règle, nous avons un nœud service (s) de contrôle : PaaS ou SaaS, lié à un nom interne (x). La réplication est effectuée dans le côté droit, dont la nouvelle instance du service (s') est déployée, et aussi liée au nom interne (x).

La règle R6 définit le déploiement d'un nouveau service dans le cloud. Elle consiste à rajouter un nouveau nœud service au sein d'une plateforme ou infrastructure. Sa forme algébrique est définie comme suit :

$$R6 :: o.(dc.(p.(v))) \rightarrow o.(dc.(p.(v.(s))))$$

Cette règle peut être appliquée aux trois types de services cloud tout en préservant les contraintes d'inclusion entre eux.

La règle R7 permet la suppression des instances inutilisées de services cloud, et qui ont été générées par la règle de réplication "R5" ou celle du déploiement "R6". Elle représente le "scaling-down" au niveau service. Sa forme algébrique est donnée comme suit :

$$R7 :: o.(dc.(p.(v.(s)))) \rightarrow o.(dc.(p.(v)))$$

Cette règle peut s'appliquer aux trois types de services, à condition que les nœuds de contrôle "IaaS" ou "PaaS" soient des nœuds vides.

Reconfiguration de la Couche Virtualisation

Les serveurs virtuels peuvent à tout moment migrer d'un serveur physique à un autre, ils peuvent aussi être répliqués ou déployés suite à la montée en charge ou la descente en charge dans le cloud. Nous définissons un ensemble de règles de réaction pour la prise en charge de cette dynamique liées à la couche virtualisation (bigraphe CVB) ;

La règle R8 exprime le fait qu'un serveur virtuel peut changer sa localité d'un centre de donnée à un autre. Cette règle est algébriquement exprimée comme suit :

$$R8 :: o.(dc_e.(p.(v)))||o'.(dc'_e.(p')) \rightarrow o.(dc_e.(p))||o'.(dc'_e.(p'.(v')))$$

Elle ne peut être appliquée qu'après l'établissement d'un lien (connexion réseau) entre les deux centres de données.

La règle R9 spécifie la réplication d'un serveur virtuel dans d'autres centres de données. La forme algébrique associée à cette règle est comme suit :

$$R9 :: o.(dc.(p.(v)))||o'.(dc'.(p')) \rightarrow o.(dc.(p.(v)))||o'.(dc'.(p'.(v')))$$

Dans le côté gauche de la règle, le centre de données (dc) contient un serveur virtuel (v). Alors que pour le côté droit, le serveur virtuel (v) est répliqué par un nouveau serveur virtuel (v') au sein du centre de données (dc').

La règle R10 décrit le déploiement d'un nouveau serveur virtuel au sein d'un serveur physique. Sa forme algébrique est donnée comme suit :

$$R10 :: o.(dc.(p)) \rightarrow o.(dc.(p.(v)))$$

Cette règle consiste à créer un nouveau nœud (v) de contrôle "VS" à l'intérieur du nœud (p) de contrôle "PS".

La règle R11 permet la suppression des images de serveurs virtuels inexploitées et accumulées suite à l'application de la règle "R9" de réplication ou celle du déploiement "R10". Elle représente le "scaling-down" au niveau virtualisation. La forme algébrique associée à cette règle est comme suit :

$$R11 :: o.(dc.(p.(v))) \rightarrow o.(dc.(p))$$

Cette règle est uniquement applicable sur les serveurs virtuels, à condition que les nœuds associés à ces serveurs doivent être vides avant leur suppression.

La règle R12 représente la consolidation de serveurs ; le processus de migration des serveurs virtuels vers un seul serveur physique [Hausman et al., 2013]. Cette règle a pour objectif la réduction de la consommation d'électricité en mettant hors tension les serveurs physiques inactifs (après migration des serveurs virtuels). La représentation algébrique de cette règle est donnée comme suit :

$$R12 :: o.(dc.(p.(v))|p'.(v'))) \rightarrow o.(dc.(p.(v|v')))$$

Dans le côté gauche de la règle, deux serveurs physiques représentés par les nœuds (p et p') de contrôle "PS", exécutent respectivement les deux serveurs virtuels modélisés par les nœuds (v et v') de contrôle "VS". Alors que dans le côté droit, le serveur virtuel (v') est migré vers le serveur physique (p), ainsi le serveur physique (p') est mis hors tension (nœud supprimé).

Exemple

En raison d'intérêts financiers (par exemple, réduire la consommation d'électricité), l'administrateur du laboratoire d'analyse décide de déplacer son serveur virtuel (local) vers un fournisseur de services cloud. Ce scénario peut être effectué en utilisant la règle "R8" (voir tableau 4.4) ; cette règle nécessite au préalable qu'une connexion entre les centres de données concernés (dc1 et dc2) soit établie. L'application de la règle de réaction bigraphique (R8) pour notre cas, est schématisée dans la figure 4.9.

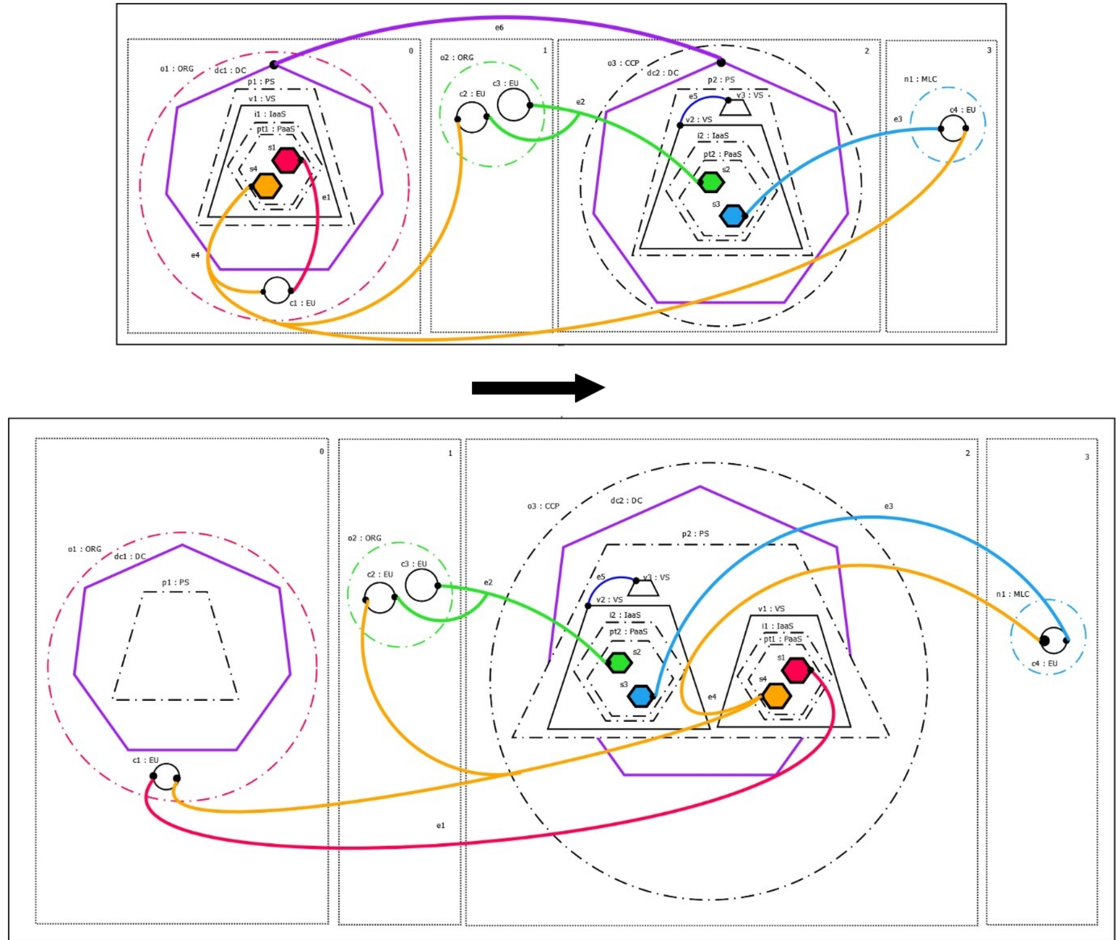


FIGURE 4.9 – Exemple d'application de la règle de réaction R8

Cette figure représente un scénario de fédération dans le cloud, consistant à migrer un serveur virtuel d'un centre de données vers un autre. Dans le côté gauche de la règle, le serveur virtuel (v1) contenant les services (S1 et S4) est déployé à l'intérieur du centre de données (dc1), au sein des locaux du laboratoire d'analyse

médicale (O1). Dans le coté droit, le serveur virtuel (v1) est migré vers le centre de données (dc2), qui est hébergé dans les locaux du fournisseur cloud (O3). Ainsi, la forme algébrique modélisant cet exemple est comme suit :

$$o1.(dc1_{e6}.(p1.(v1.(i1.(pt1.(s1_{e1}|s4_{e4}))))))|c1_{e4,e1})||$$
$$o2.(c2_{e4,e2}|c3_{e2})||o3.(dc2_{e6}.(p2.(v2_{e5}.(i2.(pt2.(s2_{e2}|s3_{e3})))|v3_{e5})))||n1.(c4_{e3,e4})$$
$$\rightarrow$$
$$o1.(dc1.(p1)|c1_{e4,e1})||o2.(c2_{e4,e2}|c3_{e2})||$$
$$o3.(dc2.(p2.(v1.(i1.(pt1.(s1_{e1}|s4_{e4})))|v2_{e5}.(i2.(pt2.(s2_{e2}|s3_{e3})))|v3_{e5})))||n1.(c4_{e3,e4})$$

Il convient de noter que d'autres scénarios de reconfigurations peuvent être identifiés, conformément aux règles de réaction proposées.

4.7 Conclusion

Dans ce chapitre, nous avons adopté une architecture en trois couches pour décrire les systèmes cloud. Cette vue offre un support méthodologique permettant de décomposer un système cloud en un ensemble d'éléments compréhensibles, chacun s'intéressant à une seule préoccupation (application du concept génie logiciel : "la séparation des préoccupations"). Ensuite, nous avons proposé un modèle formel basé bigraphes pour la spécification structurelle de cette architecture. En particulier, cette proposition met en avant trois bigraphes élémentaires : premièrement le CUB (Cloud Users Bigraph) formalisant l'ensemble des concepts de la couche utilisateurs cloud. Ensuite, le CSB (Cloud Services Bigraph) issue des différents éléments de la couche services cloud. Enfin, le CVB (Cloud Virtualization Bigraph) constitue la formalisation de la couche virtualisation cloud. La composition de ces trois bigraphes offre un bigraphe plus complexe ; le CAB (Cloud Architecture Bigraph), fournissant une vue globale du système cloud à spécifier. Ce bigraphe décrit les différentes relations entre les trois couches identifiées.

A travers ce chapitre, nous avons aussi proposé un ensemble de règles de réaction modélisant les reconfigurations dynamiques au niveau des trois couches cloud identifiées. Pour la couche utilisateurs cloud, nous avons spécifié la mobilité, la connexion et la déconnexion des utilisateurs dans le cloud. En ce qui concerne la couche services cloud, nous avons formalisé la migration, la réplication, le déploiement et la destruction des services. Enfin, pour la couche virtualisation cloud, nous avons proposé quatre règles pour formaliser la dynamique des serveurs virtuels, ainsi qu'une règle permettant la consolidation des serveurs physiques. Il convient notamment de noter que les règles de réaction présentées tout au long de ce chapitre constituent des méta-règles proposant un niveau d'abstraction élevé, et peuvent être instanciées pour spécifier et analyser les différents scénarios de fédérations dans le cloud.

Enfin, les bigraphes proposés dans ce chapitre (CAB, CUB, CSB et CVB) permettent uniquement de décrire les relations inter-couches de l'architecture adoptée. Ils spécifient l'interaction entre les utilisateurs et les services cloud, le déploiement de services au sein des serveurs virtuels et l'emplacement des centres de données par rapport aux utilisateurs dans le cloud. Dans le chapitre suivant, nous présentons une extension du bigraphe "CSB" qui prend en considération les relations internes de la couche services cloud.

Chapitre 5

Formalisation de la Composition des Services Cloud



"As far as the laws of mathematics refer to reality, they are not certain. As far as they are certain, they do not refer to reality" –Albert Einstein.

Table des matières

5.1 Introduction

De nos jours, les services cloud sont devenus une réalité proposée par de nombreux acteurs du marché IT (Amazone, Google, Oracle, IBM, etc.). Cette nouvelle génération de services a révolutionné la façon dont l'utilisateur interagit avec les applications qu'il utilise. En effet, les applications traditionnelles ne sont plus installées sur les machines épuisées de l'utilisateur, elles sont plutôt fragmentées en un ensemble de services cloud, qui sont ensuite hébergés à travers Internet sur des serveurs puissants. Cette approche représente une véritable alternative en matière de coût et de flexibilité pour les entreprises. Elle suscite l'intérêt des chercheurs, qui visent à tirer profit de la fragmentation des applications en services pour créer des services complexes combinant ceux qui existent déjà. La composition des services cloud permet de créer un service composé en exprimant les nouvelles fonctionnalités requises par les utilisateurs cloud. Elle fait face à de nombreuses contraintes et questions soulevées [Jula et al., 2014]. Une des questions pertinentes dans ce contexte, est de savoir comment modéliser les services cloud pour supporter la composition avec toutes ses facettes. Face à cette problématique, nous contribuons par la proposition d'un modèle basé bigraphes pour la modélisation de la composition des services cloud. Nous reprenons l'architecture cloud proposée dans le chapitre précédent en s'intéressant plus particulièrement à la couche *"services cloud"* avec une vue plus raffinée, permettant de prendre en considération les différents aspects liés à la composition de services cloud. Par conséquent, un modèle basé bigraphes (CScB –"Cloud Services composition Bigraph") est proposé dans ce chapitre pour la spécification de la composition des services cloud ; en présentant une vue plus détaillée de la couche correspondante (CSB).

Nous présentons dans la section 2 un exemple explicitant et motivant la composition des services cloud. Ensuite, dans la section 3, nous proposons un méta-modèle décrivant les différents concepts proposés autour de la composition des services cloud. Ceci fournira une base préalable pour la définition d'un modèle formel basé bigraphes, qui sera présenté dans la section 4. Enfin, la notion de règle de réaction bigraphique est exploitée dans la section 5 pour formaliser la composition dynamique des services cloud.

5.2 Exemple de Motivation : "$Cloud - ERS$"

Le nombre de décès de la circulation routière est une tragédie mondiale avec une tendance à hausse inacceptable ; plus de 1,25 million de personnes au monde sont tuées sur les routes chaque année [World-Health-Organization, 2015]. Apporter une assistance médicale aux victimes d'accidents de la route dans la première heure (l'heure d'or en cas d'urgence), sert comme un moyen important pour réduire la

gravité des blessures, voire même sauver des vies humaines. Pour cette raison, l'amélioration des systèmes d'intervention d'urgence existants est également un bon moyen pour réduire le nombre de morts dû aux accidents de la route.

Le Cloud Computing ouvre de nouvelles opportunités pour les différents types d'entreprises, tels que : e-santé, e-commerce, e-gouvernement et e-environnement. En particulier, l'adoption de ce paradigme dans le domaine de la santé permet d'avoir un impact important sur l'amélioration des systèmes d'intervention d'urgence existants. Effectivement, les services cloud avec leurs caractéristiques principales (accessibilité, disponibilité et élasticité) offrent une excellente solution pour diminuer le temps de réponse de ces systèmes ; ce qui représente une occasion pour un accès rapide et une assistance immédiate dans le cas d'un accident de la route.



FIGURE 5.1 – Étude de cas : Cloud-ERS

Les systèmes d'intervention d'urgence basés cloud (Cloud-ERS –"Cloud Emergency Response Systems") fournissent aux équipes d'intervention (ambulances, pompiers, police, etc.) des informations précises et pertinentes sur l'urgence de la situation (la gravité des blessures ou les dommages sur la voiture), et la localisation de la voiture accidentée. Cependant, vu leur hétérogénéité, la modélisation de ces systèmes représente un problème typique de la composition des services cloud (voir la figure 5.1).

Cette étude de cas concrète sera utilisée à travers ce chapitre pour illustrer les différents concepts liés à la proposition d'un modèle basé bigraphes pour la com-

position des services cloud.

5.3 Méta-modèle pour la Composition des Services Cloud

La composition des services Cloud, similaire à la composition des services Web, représente la capacité de construire et d'offrir de nouveaux services qui sont plus complexes en réutilisant des services de base. Le but est de créer de nouvelles fonctionnalités. Cependant, il existe plusieurs aspects permettant la distinction des services Cloud par rapport aux services Web.

5.3.1 Définition d'un Service Cloud Composite

Nous relevons dans ce qui suit une définition claire d'un service cloud composite à travers un méta-modèle. Particulièrement, les services Cloud sont conçus pour fournir un accès *évolutif* aux ressources informatiques. Ils sont généralement classés selon deux vues [Mell and Grance, 2011] : (1) la manière dont ils ont été livrés ("IaaS", "PaaS" et "SaaS"), et (2) la façon dont ils ont été déployés (public, privé, communautaire et hybride). Compte tenu de la nature des services cloud, nous suggérons qu'un service cloud composite peut être obtenu selon deux types de composition distincts mais complémentaires ; nous appelons le premier : *"Composition Verticale"*, et le deuxième : *"Composition Horizontale"*. La composition verticale offre une vue panoramique de l'imbrication hiérarchique entre les multiples services cloud. Elle se réfère aux trois modèles de services ("IaaS", "PaaS" et "SaaS"), dont les services de plus haut niveau reposent généralement sur les niveaux inférieurs de la hiérarchie. La composition horizontale, d'autre part, décrit comment les multiples services cloud, d'un niveau donné, peuvent interagir par échange de messages (demandes d'opérations) pour construire des services plus complexes. Ce second type de composition est plus proche de la composition des services traditionnels (web), avec la particularité qu'il tire partie des quatre modèles de déploiement des services cloud.

En vue de clarifier et unifier les notions précitées, nous avons eu recours à une solution indépendante de toute plateforme ou technologie particulière. Nous proposons un méta-modèle abstrait pour la capture des différentes notions du service cloud composite. Le méta-modèle met en avant les deux facettes de la composition des services cloud : la *"Composition Verticale"* et la *"Composition Horizontale"*. Ce méta-modèle peut être transformé par la suite vers des spécifications exécutables, ou bien utilisé pour la génération de code.

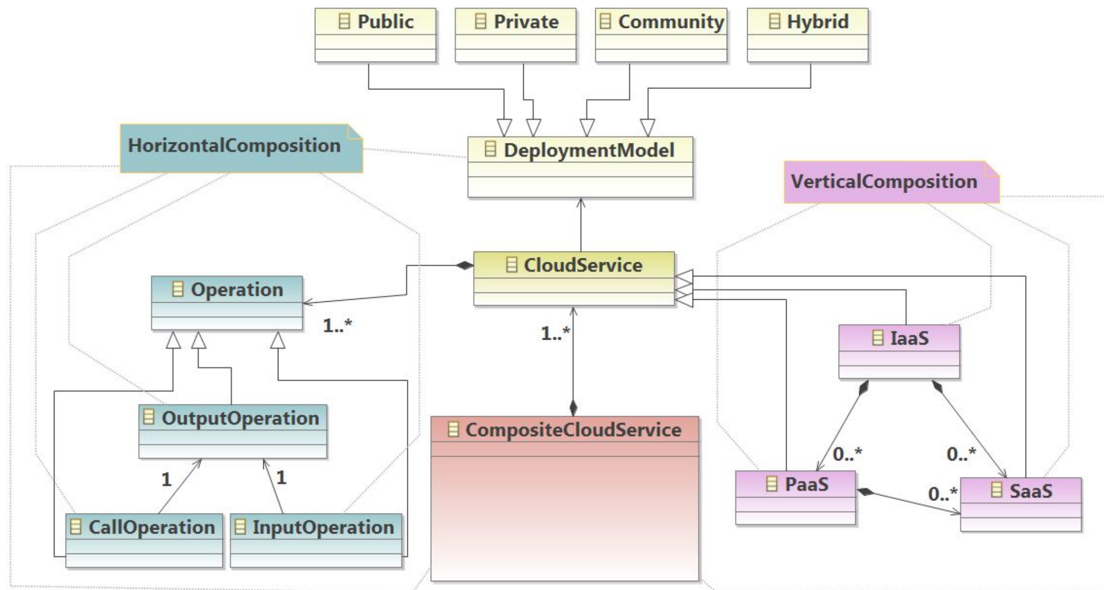


FIGURE 5.2 – Méta-modèle pour la composition des services Cloud

La figure 5.2 illustre notre méta-modèle abstrait, qui comporte quatorze méta-classes impliquées dans la composition des services cloud. Le concept noyau de ce méta-modèle est une méta-classe nommée*"CompositeCloudService"*, qui est utilisée pour modéliser la capacité d'un service cloud à être composé d'autres services. Elle possède une agrégation de type *"un à plusieurs"* où la méta-classe *"CloudService"* est la cible. Cette méta-classe *"CompositeCloudService"* encapsule également les deux facettes de la composition des services cloud : *"VerticalComposition"* et *"HorizontalComposition"*. Alors que la première *"VerticalComposition"* a une dépendance vers les modèles de services cloud : *"IaaS", "PaaS" et "SaaS"*, la deuxième *"HorizontalComposition"* tire partie des différents types d'opérations : *"OutputOperation"*, *"CallOperation"* et *"InputOperation"*. D'autre part, la méta-classe *"CloudService"* relie trois sous-méta-classes spécifiant les trois niveaux d'abstraction (modèles de prestation) des services cloud, à savoir : *"IaaS"* pour l'infrastructure en tant que service, *"PaaS"* pour la plateforme en tant que service, et enfin,*"SaaS"* pour l'application en tant que service. Ces trois méta-classes (*"IaaS", "PaaS" et "SaaS"*) ont à leur tour trois agrégations exprimant les conditions d'imbrication entre les différents niveaux de services cloud. La méta-classe *CloudService"* a aussi une dépendance (*déployé en tant que*) vers la méta-classe *"DeploymentModel"*, qui à son tour relie quatre sous-méta-classes modélisant les modèles de déploiement cloud, à savoir, *"Public", "Privé", "Communautaire" et "Hybride"*. Enfin, la méta-classe *"Operation"* collabore avec ses trois sous-méta-classe (*"InputOperation", "CallOperation" et "OutputOpration"*)

dans un mécanisme de haut niveau pour l'interprétation du comportement de la méta-classe *"CloudService"*.

5.3.2 Exemple d'instanciation

Pour illustrer la composition horizontale des services cloud, nous nous référons à l'exemple précédent. Ainsi, nous identifions le service composite suivant : *"Emergency Response Cloud Service"* (*ERCS*) qui signale un accident de la route à une position donnée (*reportAccident()*). Ce service interagit (en termes de demandes d'opérations) avec quatre services existants : *"Hospital Cloud Service" (HCS), "Firefighter Cloud Service" (FCS), "Police Cloud Service" (PCS) and "Automotive-garage Cloud Service" (ACS).* Le service (*HCS*) permet d'envoyer une ambulance vers l'endroit de l'accident identifié par la position GPS donnée (*sendAmbulanceSquad ()*), et aussi de créer un rapport sur la gravité des blessures (*createHealthRecord ()*). Le service *(FCS)* permet d'envoyer un camion de pompiers à la position du véhicule accidenté (*sendFirefightingApparatus ()*). Le service *(PCS)* offre la possibilité de mettre à jour la carte de circulation routière en fonction de l'emplacement de l'accident signalé (*updateTrafficMap ()*), et permet également d'envoyer une patrouille de police (*sendPatrolCar ()*). Enfin, le service *(ACS)* permet d'envoyer le bon dépanneur en fonction des informations concernant les dommages causés sur la voiture (*sendTowTruck ()*).

D'un autre coté, nous considérons, grâce à ce méta-modèle, la deuxième facette de composition des services cloud. Ainsi, les services cloud précités (*ERCS, HCS, FCS, PCS et ACS*) doivent faire appel à des services cloud de niveau inférieur de la hiérarchie, c.-à-d., PaaS et IaaS. Nous considérons alors les cinq nouveaux services cloud suivants : *"Google Compute Engine" (GCE), "Google App Engine" (GAE), "IBM SoftLayer", "IBM BlueMIX" and "Amazon Elastic Compute" Cloud (AEC2).* Le service *(GCE)*, est une offre "IaaS" de Google qui permet d'exécuter des programmes à grande échelle sur le matériel physique. Le service *(GAE)*, est une offre "PaaS" de Google pour le développement et l'hébergement des applications cloud. Le service *(SoftLayer)*, est une offre "IaaS" d'IBM pour construire des applications de haute performance. Le service *(BlueMIX)* est une offre "PaaS" d'IBM qui fournit des solutions hautement sécurisées et fiables. Enfin, le service *(AEC2)* est une offre "IaaS" d'Amazon qui fournit une capacité de calcul redimensionnelle dans le cloud.

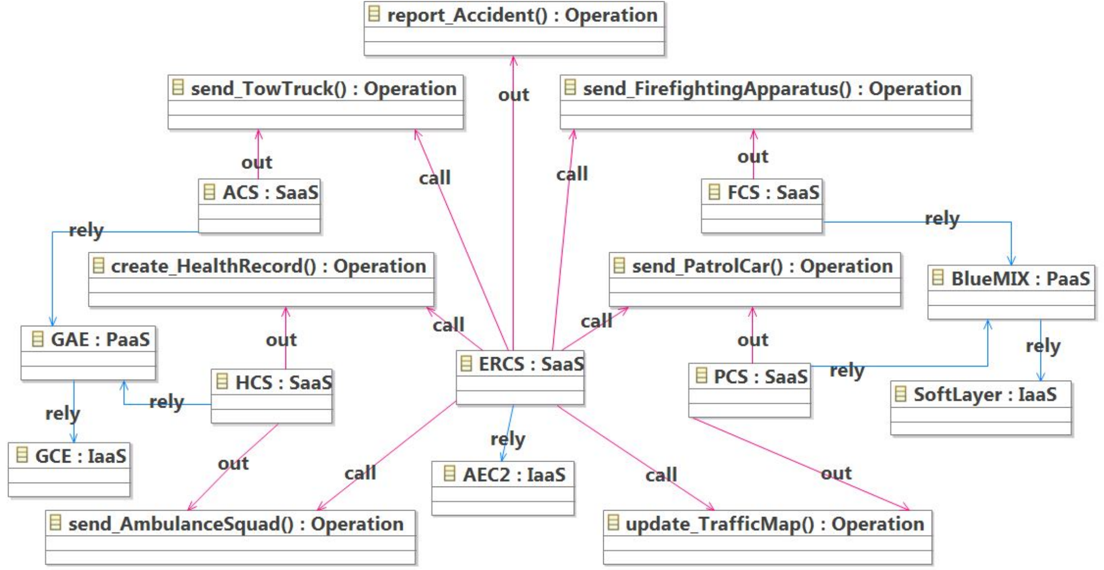


FIGURE 5.3 – Exemple d'instanciation du méta-modèle

La figure 5.3 montre une instance du méta-modèle proposé, elle représente un modèle prenant en considération les deux facettes de la composition des services cloud dans notre cas d'étude. En ce qui concerne la composition verticale, le modèle spécifie les relations "*rely*" entre les services cloud eux-même ; en tant que structure arborescente basée sur les trois niveaux de services cloud. Par exemple, le service *(ERCS)* repose sur le service *(AEC2)*, aussi le service *(HCS)* repose sur le service*(GAE)*, qui est à son tour contenu dans le service *(GCE)*. En outre, les relations "*call* et *out*" entre les services cloud et les opérations dénote dans notre cas la composition horizontale. Le service cloud composé *(ERCS)* se base sur six opérations : *sendAmbulanceSquad (), createHealthRecord (), sendFirefightingApparatus (), updateTrafficMap (), sendPatrolCar () et sendTowTruck ()*, issues des quatre services cloud existants (*HCS, FCS, PCS et ACS*) pour fournir une nouvelle fonctionnalité (*reportAccident ()*).

Le méta-modèle proposé offre aussi une vue compréhensible des concepts liées aux services cloud ; il prend en compte les deux facettes relatives à leur composition : la *Composition Verticale* et la *Composition Horizontale*. Ce méta-modèle clarifie notre vocabulaire de modélisation, il peut servir à de nombreuses applications, nous l'utilisons dans ce travail pour simplifier le passage aux modèles bigraphiques décrivant la sémantique de composition.

5.4 Sémantique basée Bigraphe pour la Composition des Services Cloud ($CScB$)

Fortement basées sur des définitions mathématiques, les méthodes formelles peuvent être appliquées afin de fournir une sémantique précise aux différents concepts du méta-modèle proposé pour un service cloud composite. La théorie des bigraphes constitue aussi une solution appropriée pour la formalisation de la composition des services cloud avec ses deux facettes (Verticale et Horizontale). Par conséquent, nous établissons dans la présente section une relation de correspondance entre la théorie des bigraphes et les concepts issus du méta-modèle proposé.

5.4.1 Principe de notre approche

A partir du méta-modèle défini pour la composition des services cloud, nous établissons un modèle bigraphique noté "CScB" qui prend en considération les deux aspects relatifs à la composition des services cloud : la *composition verticale* –formalisée à travers le graphe de places, et la *composition horizontale* –formalisée via le graphe de liens (voir le tableau 5.1). Dans ce bigraphe, nous représentons les services cloud par un ensemble de nœuds, dont les contrôles permettent de distinguer entre les trois niveaux de services : IaaS, PaaS et SaaS. Aussi, nous suggérons que chaque nœud service dispose de trois ports : public, privé et communautaire. Ces ports sont utilisés pour relier les opérations potentielles, qui sont à leur tour représentées par un ensemble de nœuds. Le contrôle attaché à chaque nœud opération permet la spécification de son état, que ce soit un appel, un paramètre d'entrée ou bien un paramètre de sortie. La définition suivante (5.1) décrit tous les éléments mentionnés ci-dessus.

TABLE 5.1 – Correspondances entre la composition des services cloud et les bigraphes

Concepts du Méta-modèle	**Éléments Bigraphiques**
CloudService – IaaS – PaaS – SaaS	Nœud – Contrôle – Contrôle – Contrôle
Operation – InputOperation – OutputOperation – CallOperation	Nœud – Contrôle – Contrôle – Contrôle
DeploymentModel – Public – Private – Community	Ensemble de ports – Port – Port – Port
CompositeCloudService –VerticalComposition –HorizontalComposition	Bigraphe – Graphe de places – Graphe de liens
Association "CloudService–Operation"	Arc entre les nœuds Service et Opération
Association "CloudService–CompositeCloudService"	Typage des places et liens (place-link sorting)
Associations "IaaS–PaaS–SaaS"	Règles de formation sur le graphe de places
Associations "InputOperation–OutputOperation–CallOperation"	Règles de formation sur le graphe de liens

Définition 5.1 ***Bigraphe pour la Composition des Services Cloud***
Le bigraphe formalisant la composition des services cloud prend la forme :

$$CScB = (V_{CScB}, E_{CScB}, ctrl_{CScB}, prnt_{CScB}, link_{CScB}) \equiv < GP_{CScB}, GL_{CScB} > : \\ < m_{CScB}, X_{CScB} > \rightarrow < n_{CScB}, Y_{CScB} > ,$$

dont :

- V_{CScB} *représente l'ensemble de nœuds services et opérations ;*
- E_{CScB} *est l'ensemble finie d'hyper-arcs, connectant un nœud service à un nœuds opération ;*
- $ctrl_{CScB} : V_{CScB} \rightarrow S_{CScB}$ *est la transformation de contrôle, où la signature* S_{CScB} *est définie par* $S_{CScB} = \{IaaS, PaaS, SaaS, InputOp, CallOp, OutputOp\}$. *De plus, la transformation* $ar : S_{CScB} \rightarrow N$ *attribue une arité (un certain nombre de*

ports) à chaque contrôle, tel que $ar(IaaS) = ar(PaaS) = ar(SaaS) = 3, ar(InputOp) = ar(OutputOp) = 2$ *et* $ar(CallOp) = 1$ *;*
— X_{CScB} *est l'interface interne et* Y_{CScB} *est l'interface externe ;*
— m_{CScB} *est le nombre de sites, et* n_{CScB} *représente le nombre de racines ;*
— $prnt_{CScB} : m_{CScB} \uplus V_{CScB} \rightarrow V_{CScB} \uplus n_{CScB}$ *est la transformation de parenté, spécifiant le parent de chaque nœud ;*
— $link_{CScB} : X_{CScB} \uplus P_{CScB} \rightarrow E_{CScB} \uplus Y_{CScB}$ *est la transformation de liaison, avec* P_{CScB} *un ensemble de ports ;*
— GP_{CScB} *est le graphe de places spécifiant la composition verticale des services cloud, alors que leurs composition horizontale est représentée via le graphe de liens* GL_{CScB}*.*

Le CScB proposé (dans la définition 5.1) spécifie la composition des services cloud via ses deux structures indépendantes : le graphe de places (GP_{CScB}) et le graphe de liens (GL_{CScB}). Plus précisément, la composition verticale des services cloud est représentée via le placement d'un nœud au sein du graphe de place, tandis que la composition horizontale est codée par les hyper-arcs du graphe de liens. Néanmoins, certaines contraintes sur la structure du bigraphe de composition CScB doivent être définies afin de contrôler le processus de composition des services et d'assurer la cohérence du bigraphe ainsi obtenu. Nous raffinons la définition du graphe de places et graphe de liens dans les sous-sections suivantes en considérant ce type de contraintes. Nous exploitons la discipline de typage des liens et places ("place-link Sorting") offerte par la théorie des bigraphes [Milner, 2009], pour contraindre la structure du bigraphe "CScB" à l'aide d'un ensemble de règles de formation.

5.4.2 Composition Verticale

Conformément à la relation d'imbrication hiérarchique des différents niveaux de services cloud, nous étendons la définition du bigraphe proposé (voir définition 5.1) pour affiner la spécification de la composition verticale. Pour cela, nous utilisons la discipline de typage des places (*place-sorting*) afin de contraindre la transformation de parenté ($prnt_{CScB}$). Plus explicitement, nous associons des types aux contrôles (S_{CScB}) au moyen de sortes (Θ^P_{CScB}) et nous déterminons un ensemble de règles de formation (Φ^P_{CScB}).

La définition 5.2 spécifie le typage de places (*place-sort*) pour la structure du CScB.

Définition 5.2 ***Typage de place du CScB***
Chaque graphe de places GP_{CScB}*, représentant la composition verticale des services cloud, est contraint par le typage de place (place-sort) suivant :*

$$\sum^{P_{CScB}} = (\Theta^{P}_{CScB}, S_{CScB}, \Phi^{P}_{CScB})$$

où Θ^{P}_{CScB} est un ensemble non vide de sortes, S_{CScB} est une signature et Φ^{P}_{CScB} est un ensemble de règles de formation. Étant donné l'ensemble de sortes Θ^{P}_{CScB} = s, p, où s représente un service cloud et p exprime une opération. La signature S_{CScB}= IaaS : s, PaaS : s, SaaS : s, InputOp : p, CallOp : p, OutputOp : p classifie les contrôles au moyen des deux sortes. Les règles de formation $\Phi^{P}_{CScB} = \varphi 1, \varphi 2, \varphi 3, \varphi 4$ du graphe de place GP_{CScB}, sont définies comme suit :

Condition φ1 *: un IaaS-nœud a zéro ou plus de s-fils dont leur contrôle $c \in S_{CScB}$ et $c = PaaS$;*

Condition φ2 *: un IaaS-nœud peut avoir s-fils dont leur contrôle $c \in S_{CScB}$ et $c = SaaS$;*

Condition φ3 *: un PaaS-nœud a zéro ou plus s-fils dont leur contrôle $c \in S_{CScB}$, et $c = SaaS$;*

Condition φ4 *: tous les SaaS-nœuds sont atomiques ;*

Condition φ5 *: tous les p-nœuds sont atomiques.*

Un récapitulatif des contrôles, sortes et règles de formation, est présenté dans le tableau 5.2. La formalisation souligne que tous les contrôles doivent appartenir à l'ensemble de sortes Θ^{P}_{CScB}. Nous utilisons la sorte s= IaaS, PaaS, SaaS pour coder un service cloud, que ce soit une infrastructure, une plateforme ou une application en tant que service. Nous affectons aussi la sorte p = InputOp, CallOp, OutputOp aux contrôles codant l'état d'une opération. Par ailleurs, les conditions des règles de formation $\varphi 1, ..., \varphi 5$ restreignent la structure hiérarchique des nœuds services et opérations. Par conséquent, les nœuds de contrôle IaaS et PaaS sont "actifs", tandis que les nœuds de contrôle : SaaS, InputOp, CallOp et OutputOp sont "atomiques". Pour plus de clarté, nous proposons une notation graphique appropriée pour chaque contrôle (voir le tableau 5.2).

TABLE 5.2 – Typage de places du CScB

Contrôle	Activité	Typage de place	Représentation graphique	Règles de formation
IaaS	oui	s		Condition $\varphi 1$ Condition $\varphi 2$ Condition $\varphi 3$ Condition $\varphi 4$
PaaS	oui	s		
SaaS	non	s		
InputOp	non	p		Condition $\varphi 5$
CallOp	non	p		
OutputOp	non	p		

5.4.3 Composition Horizontale

Dans cette sous-section, nous montrons l'application de la discipline de typage de liens ("link-sorting") pour affiner la spécification de la composition horizontale en considérant les modèles de déploiement de services cloud.

La discipline de typage de liens ajoutée au graphe de liens (GL_{CScB}) fournira une spécification cohérente de la composition horizontale. Ce résultat est obtenu en classifiant les ports au moyen de sortes, et en limitant les liens ($link_{CScB}$) permis entre les différents ports d'un nœud (dans V_{CScB}). La définition suivante 5.3 contraint la structure du CScB afin de représenter plus explicitement la composition horizontale des services cloud.

Définition 5.3 ***Typage de liens du CScB***

Pour chaque graphe de liens GL_{CScB} formalisant la composition horizontale des services cloud, nous enrichissons la signature par le typage de liens suivant :

$$\sum\nolimits_{CScB}^{L} = (\Theta_{CScB}^{L}, S_{CScB}, \Phi_{CScB}^{L})$$

où Θ_{CScB}^{L} est un ensemble non vide de sortes, S_{CScB} est une signature et Φ_{CScB}^{L} est un ensemble de règles de formation. Étant donné que l'ensemble de sortes $\Theta_{CScB}^{L} =$

$\{pbc, prv, cmt, sto, oto\}$, sachant que pbc est pour "public", prv est pour "privé", cmt est pour "communautaire", sto est pour "service-à-opération" et oto est pour "opération-à-opération". La signature S_{CScB}, classifiant chaque port à travers le typage de liens proposé, est donnée par S_{CScB} = IaaS : (isp1 :pbc, isp2 :prv, isp3 :cmt) , PaaS : (psp1 :pbc, psp2 :prv, psp3 :cmt) , SaaS : (ssp1 :pbc, ssp2 :prv, ssp3 :cmt) , InputOp : (iop1 :sto, iop2 :oto) , CallOp : (cop1 :sto) , OutputOp : (oop1 :sto, oop2 :oto) , avec : isp_i est le i-ème port du IaaS-nœud, psp_i est le i-ème port du PaaS-nœud, ssp_i est le i-ème port du SaaS-nœud, iop_i est le i-ème port du InputOp-nœud, cop_i est le i-ème port du CallOp-nœud, et oop_i est le i-ème port du OutputOp-nœud ; où : $i \in ar(S_{CScB})$. Les règles de formation $\Phi^L_{CScB} = \varphi'1, \varphi'2, \varphi'3, \varphi'4$ du graphe liens GL_{CScB}, sont définies comme suit :

Condition $\varphi'1$ *: dans un s-nœud, trois ports (pbc, prv et cmt) peuvent être lié à un p-nœud ;*

Condition $\varphi'2$ *: dans un InputOp-nœud, le port (sto) peut être relié à un s-nœud et l'autre (oto) peut être reliés à un OutputOp-nœud ;*

Condition $\varphi'3$ *: dans un OutputOp-nœud, le port (sto) peut être relié à un s-nœud et l'autre (oto) peut être relié à un InputOp-nœud ;*

Condition $\varphi'4$ *: un CallOp-nœud ne peut être lié qu'aux s-nœuds par l'intermédiaire du port (sto).*

Les sortes de port et les règles de formation utilisées pour représenter la composition horizontale (de la définition 5.3) sont résumées dans le tableau 5.3. Nous avons défini trois sortes de ports pour les nœuds services (ar (s-nœuds) = 3), dont le premier port est public (pbc), le deuxième est privé (prv) et le troisième est communautaire (cmt). Ces différents ports reflètent les trois modes d'accès (modèles de déploiement) aux services cloud. De plus, nous avons aussi spécifié deux sortes de ports pour les nœuds opérations : le premier port est service-à-opération (sto) et le deuxième est opération-à-opération (oto). Alors que les nœuds *InputOp* et *OutputOp* possèdent deux sortes de ports (ar (InputOp-node) = ar (OutputOp-noeud) = 2), les *CallOp-nœuds* ne possèdent qu'un seul port (ar (CallOp-nœud) = 1) de sorte service-à-opération (sto). Enfin, les conditions des règles de formation $\varphi'1, ..., \varphi'5$ limitent les liens entre ces différents ports ; elles indiquent que si deux ports peuvent être connectés via un lien ou non. Une notation graphique est également suggérée pour chaque sorte de port (voir le tableau 5.3).

Table 5.3 – Typage de liens du CScB

Contrôle	Arité	Typage de lien	Représentation graphique	Règles de formation
IaaS, PaaS et SaaS	3	pbc		Condition φ'1
	3	prv		
	3	cmt		
InputOp et OutputOp	2	oto		Condition φ'2 Condition φ'3
	2	sto		
CallOp	1	sto		Condition φ'4

5.4.4 Exemple d'illustration

Pour illustrer les aspects fondamentaux du CScB, nous appliquons notre approche de modélisation de la composition des services cloud sur l'exemple d'étude de cas ; le système d'intervention d'urgence basé cloud (Cloud-ERS). La figure 5.4 montre la structure statique de ce système (Cloud-ERS), le bigraphe résultant prend en charge la composition verticale et horizontale des services cloud. Cela peut être fait via les deux structures indépendantes qui partagent l'ensemble des nœuds (services et opérations), à savoir : le graphe de places (comme le montre la figure 5.4 . b) spécifiant la composition verticale, et le graphe de liens (comme le montre la figure 5.4 .c) spécifiant la composition horizontale.

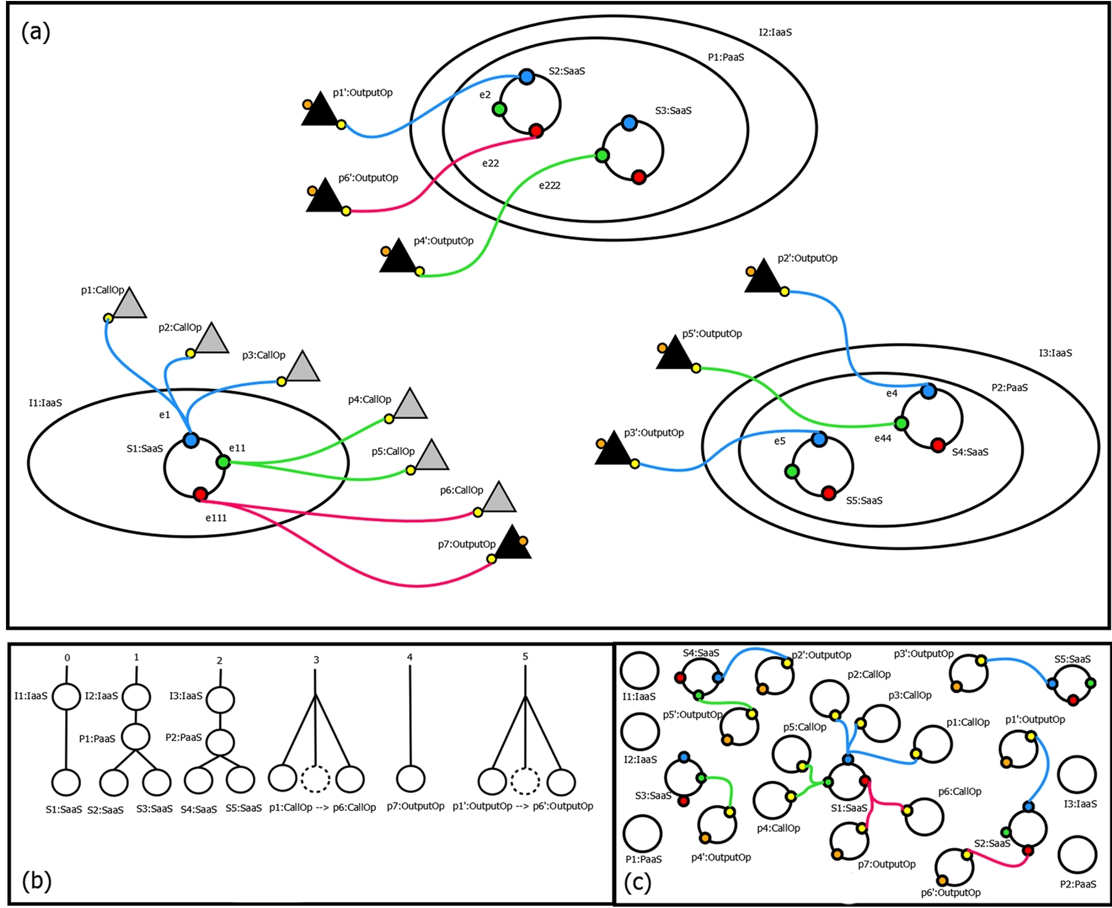


FIGURE 5.4 – Bigraphe modélisant Cloud-ERS

Selon notre approche de formalisation (le modèle CScB), nous identifions les services cloud suivants :

— $I1, I2, I3 \in V_{CScB}$ dont le contrôle est défini par : $ctrl_{CScB}(I1) = ctrl_{CScB}(I2) = ctrl_{CScB}(I3) =' IaaS'$. Ces trois services représentent respectivement : le AEC2 (Amazon Elastic Compute Cloud), le GCE (Google Compute Engine) et l'IBM SoftLayer ;

— $P1, P2 \in V_{CScB}$ de contrôle $ctrl_{CScB}(P1) = ctrl_{CScB}(P2) =' PaaS'$, représentant respectivement le GAE (Google App Engine) et IBM BlueMIX ;

— $S1, S2, S3, S4, S5 \in V_{CScB}$ de contrôle $ctrl_{CScB}(S1) = ctrl_{CScB}(S2) = ctrl_{CScB}(S3) = ctrl_{CScB}(S4) = ctrl_{CScB}(S5) = "SaaS"$, représentant respectivement : le service ERCS (Emergency Response Cloud Service), le service HCS (Hospital Cloud Service), le service ACS (Automotive-garage Cloud Service), le service PCS (Police Cloud Service) et le service FCS (Firefighter Cloud Service).

Chacun des services cloud identifiés peut fournir ou demander des opérations différentes (voir figure 5.4). Le cloud service HCS (S2) offre deux opérations : sendAmbulanceSquad() et createHealthRecord(), codées par les nœuds (p1') et (p6') de contrôle $ctrl_{CScB}(p1') = ctrl_{CScB}(p6') = OutputOp$. Le service cloud ACS (S3) fournit une seule opération : sendTowTruck(), codée par le nœud (p4') de contrôle $ctrl_{CScB}(p4') = OutputOp$. Le service cloud PCS (S4) fournit deux opérations : sendPatrolCar() et updateTrafficMap(), codées par des nœuds (p2') et (p5') de contrôle $ctrl_{CScB}(p2') = ctrl_{CScB}(p5') = OutputOp$. Le service cloud FCS (S5) fournit une seule opération : sendFirefightingApparatus(), codée par le nœud (p3') de contrôle $ctrl_{CScB}(p3') = OutputOp$. Enfin, le service cloud composite : ERCS (S1) fournit une opération reportAccident() et utilise six opérations existantes. Ceci est respectivement codé par : le nœud (p7') de contrôle $ctrl_{CScB}(p7') = OutputOp$, et les nœuds (p1, p2, p3, p4, p5 et p6) de contrôle $ctrl_{CScB}(p1) = ctrl_{CScB}(p2) = ctrl_{CScB}(p3) = ctrl_{CScB}(p4) = ctrl_{CScB}(p5) = ctrl_{CScB}(p6) = CallOp$.

Dans cette section, nous avons défini la composition des services cloud de manière statique, nous nous intéressons dans la section suivante à modélisation de sa dynamique. La composition dynamique des services cloud permet de créer de manière *autonome* des services complexes en *combinant* les services existants à la volée en fonction des demandes de l'utilisateur et du contexte. Elle aura lieu uniquement au moment de l'exécution.

5.5 Modélisation de la Composition Dynamique des Services Cloud

La composition des services cloud étant modélisée par un bigraphe (CScB), tout changement dynamique qui altère cette composition peut être schématiser par une règle de réaction. Dont les bigraphes "Redex" et "Reactum" définissent les états de cette composition avant et après l'arrivée d'un événement déclenchant ce changement. Selon les types de composition considérés, nous définissons deux méta-règles de réaction pour la composition de services cloud : règle de réaction agissant sur la composition verticale (CVcR –"Cloud Vertical composition Rule") et règle de réaction agissant sur la composition horizontale (CHcR –"Cloud Horizontal composition Rule"). Autrement dit, ces deux méta-règles produirons des effets sur le graphe de places et le graphe de liens respectivement.

Règle de réaction CVcR

La règle de réaction verticale (CVcR) modélise le changement de localisation d'un service cloud au moment de la composition, c'est-à-dire la migration du

service d'une infrastructure à une autre, ou d'une plateforme à une autre. La forme graphique associée à cette règle est présentée dans la figure 5.5. Dans le côté gauche de la règle, le s-nœud (s1) est placé dans le s-nœud (i1), tandis que dans le côté droit, le s-nœud (s1) est déplacé vers le s-nœud (i2). Nous suggérons aussi que le nœud du service cloud (s1) soit de contrôle : "PaaS" ou "SaaS", tandis que les nœuds (i1 et i2) soient de contrôle : IaaS (voir les règles de formation : $\varphi 1$ et $\varphi 2$) ou PaaS (voir la règle de formation : $\varphi 3$).

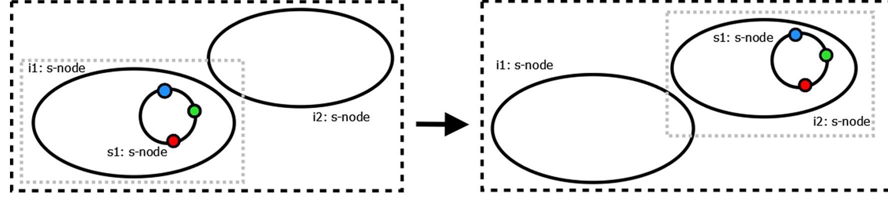


FIGURE 5.5 – Règle de réaction CVcR

La forme algébrique de cette méta-règle est la suivante :

$$i1 : IaaS{-}PaaS{-}nœud.(s1 : PaaS{-}SaaS{-}nœud) \parallel i2 : IaaS{-}PaaS{-}nœud \rightarrow$$
$$i1 : IaaS - PaaS - nœud \parallel i2 : IaaS - PaaS - nœud.(s1 : PaaS - SaaS - nœud)$$

Règle de réaction CHcR

La règle de réaction horizontale (CHcR) affecte seulement les liaisons d'un bigraphe. Sa forme graphique est présentée dans la figure 5.6, elle modélise une interaction entre deux services et l'exprime en tant que lien d'invocation d'opération. Le service cloud (s1) est relié à un autre service cloud (s2) via une demande d'opérations (p1 et p1'). Dans les deux côtés, les nœuds services (s1 et s2) peuvent être de contrôle : "IaaS", "PaaS" ou "SaaS", le nœud opération (p1') est de contrôle : "OutputOp". Aussi, le nœud opération (p1) est de contrôle : "CallOp" pour le côté gauche, tandis qu'il est de contrôle : "InputOp" dans le côté droit. Il convient également de noter que cette règle peut être appliquée aux différents ports d'un service cloud (pbc, prv ou cmt) ; ceci est modélisé à l'aide des points noirs.

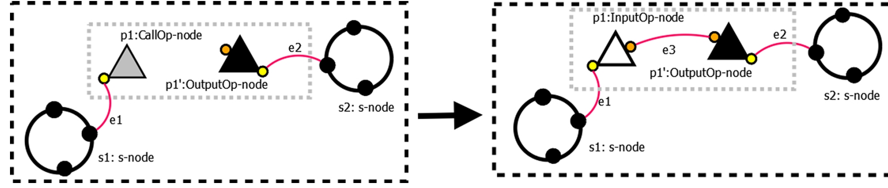


FIGURE 5.6 – Règle de réaction CHcR

La forme algébrique associée à cette règle est donnée comme suit :

$$s1e1 : s - n\oe ud \| p1e1 : CallOp - n\oe ud \parallel s2e2 : s - n\oe ud \| p1'e2 : OutputOp - n\oe ud \rightarrow s1e1 : s - n\oe ud \| p1e1, e3 : InputOp - n\oe ud \parallel s2e2 : s - n\oe ud \| p1'e2, e3 : OutputOp - n\oe ud$$

Exemple

Le résultat d'application de la règle de réaction CHcR à notre cas est donné par la figure 5.7, qui montre la sollicitation de toutes les opérations demandées (p1, p2, p3, p4, p5, p6) par le service cloud ERCS (S1), aux opérations offertes (p1 ', p2', p3 ', p4', p5 ', p6') par les services cloud existants (S2, S3, S4, S5). Ceci est effectué via l'ensemble des arcs : ea1, ea2, ea3 , ea4 , ea5 et ea6.

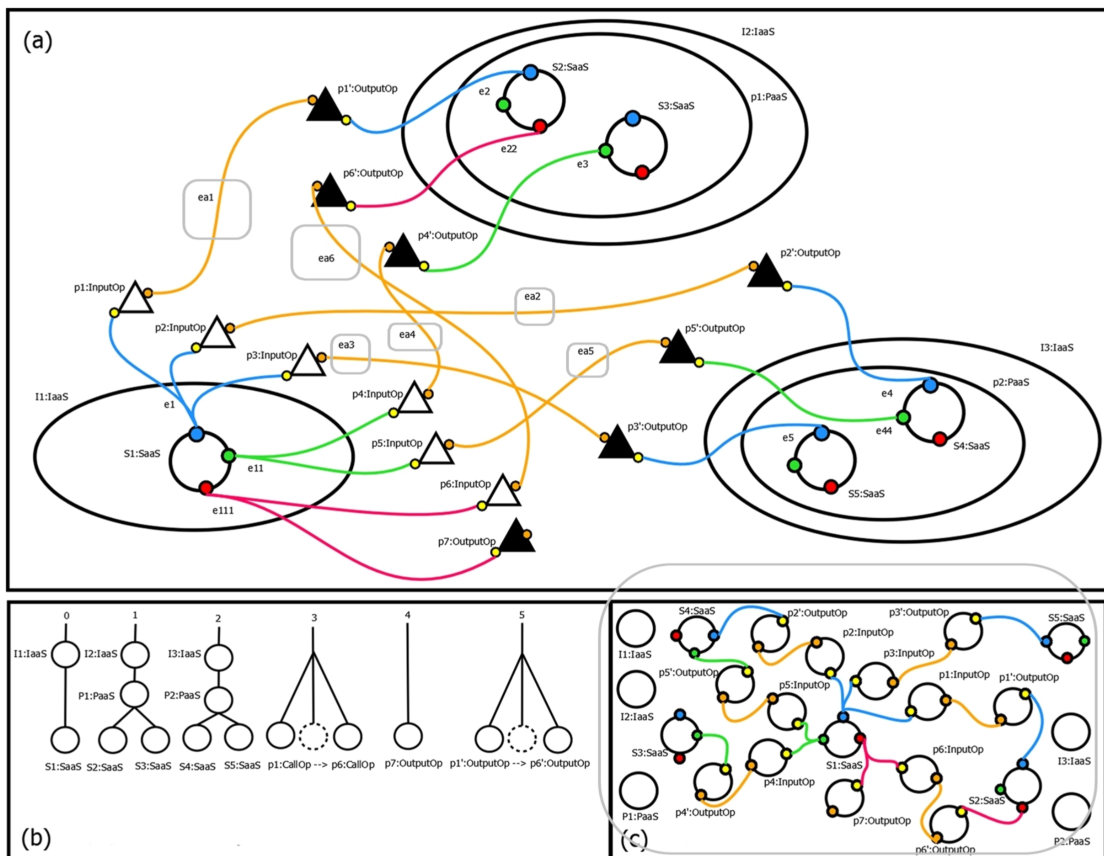


Figure 5.7 – Exemple d'application de la règle CHcR

D'autre part, et pour des raisons de sûreté, l'administrateur du service cloud HCS (S2) décide de déplacer son service de la plateforme de Google (P1) vers la plateforme d'IBM (P2). Cela peut être effectué en instanciant la méta-règle CVcR, la figure 5.8 montre le résultat de son application.

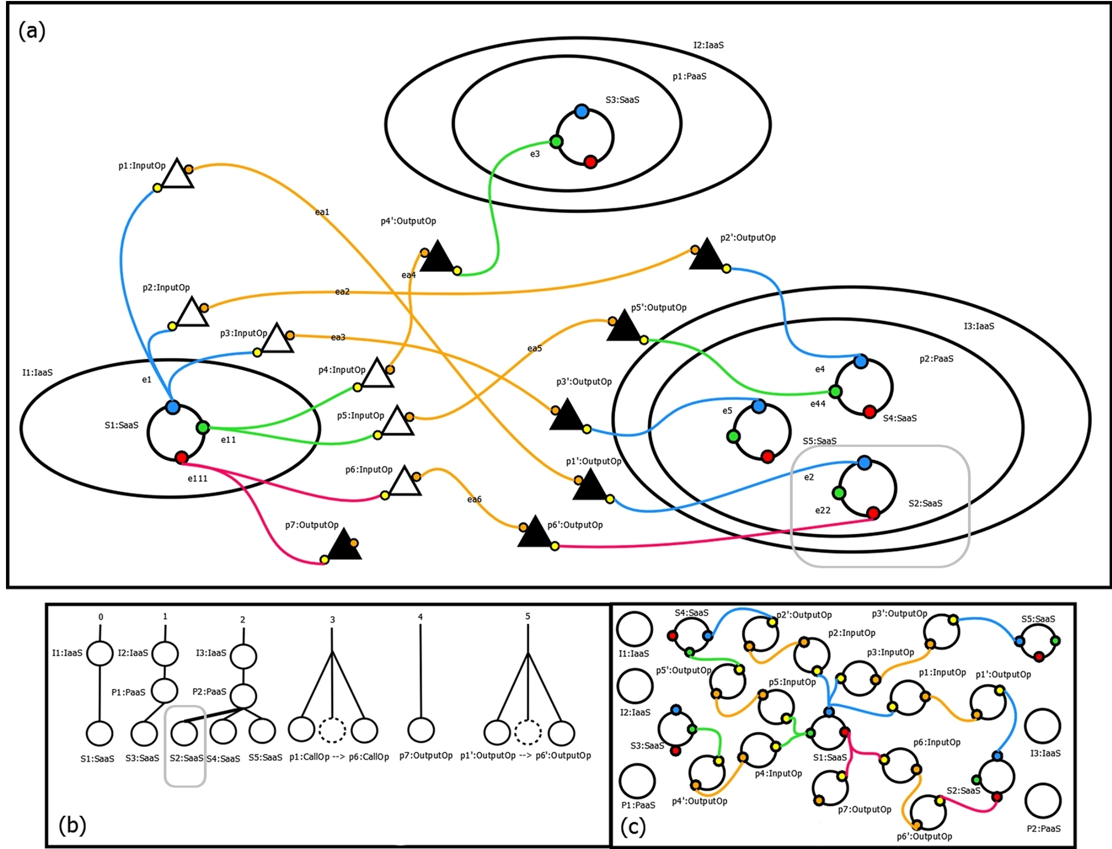


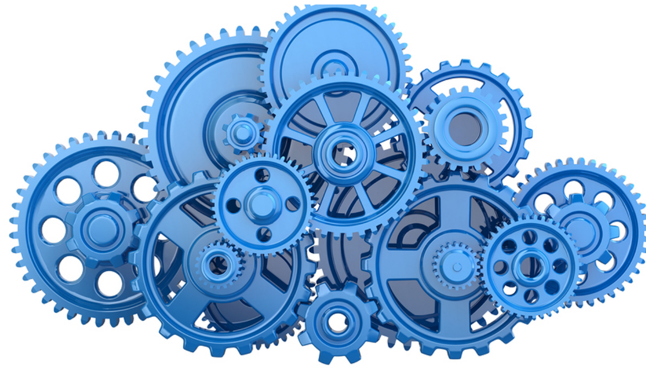
Figure 5.8 – Exemple d'application de la règle CVcR

5.6 Conclusion

A travers ce chapitre, nous avons proposé une approche formelle où la composition des services cloud peut être modélisée selon deux dimensions différentes (Verticale et Horizontale). Comme premier résultat, un méta-modèle abstrait a été proposé afin de capitaliser les concepts de base nécessaires à la composition de services cloud, ensuite nous avons associé une sémantique formelle à base des bigraphes aux concepts de ce méta-modèle. Ainsi, un modèle bigraphique pour la composition des services cloud (CScB) a été déduit. Ce modèle a été enrichi avec un ensemble de méta-règles de réaction bigraphiques (CVcR et CHcR) pour spécifier la dynamique de composition des services cloud.

Chapitre 6

"RCTool4Bigraphs" : Outil pour la Réécriture et la Vérification des Bigraphes



"The problem of verifying was a problem of simulation or data representation, and I realised how big a problem that was going to be. In fact, out of that I got interested in simulation, which I did a bit of work on" –Robin Milner.

Table des matières

6.1 Introduction

Les bigraphes représentent un formalisme récent permettant de modéliser deux aspects importants des systèmes ubiquitaires (*"Mobile Locality"* et *"Mobile Connectivity"*). En effet, un bigraphe consiste en un graphe de places ; spécifiant la distribution spatiale des nœuds, et un graphe de liens ; connectant ces différents nœuds. Ces deux graphes (graphe de places et graphe de liens) peuvent être reconfigurés via un ensemble de règles de réaction. En se basant sur la théorie des catégories, la théorie des bigraphes représente une théorie générale offrant la possibilité d'unifier les théories existantes. On peut citer pour cela, l'encodage basé bigraphe des réseaux de Petri [Milner, 2004], et celui des langages de processus concurrents (CCS, p-calcul, x-calcul, etc.)[Milner, 2005]. Par conséquent, cette théorie a été largement adoptée dans plusieurs domaines, y compris les réseaux sans fil [Sevegnani, 2012b, Boucebsi and Belala, 2015], les systèmes sensibles au contexte [Birkedal et al., 2006, Cherfia et al., 2014], les applications biologiques [Damgaard and Krivine, 2008] et les processus d'affaires [Bundgaard et al., 2008]. Rajoutant à cette liste, notre proposition concernant l'adoption des bigraphes pour le cloud computing [Benzadri et al., 2013a, Benzadri et al., 2013b, Benzadri et al., 2014b, Benzadri et al., 2014a, Benzadri et al., 2015, Benzadri et al., 2016].

Telles que soulignées par le fondateur de la théorie des bigraphes – Milner [Milner, 2009], la manipulation et l'analyse manuelle des modèles bigraphiques est une tache d'une extrême complexité. Pour palier à ce problème, l'implémentation d'outils logiciels permettant l'édition, l'exécution (la réécriture) et la vérification des bigraphes, est fortement sollicitée. En conséquence, un ensemble d'outils appropriés est proposé dans la littérature : BPL Tool [Hojsgaard and Glenstrup, 2011], Big Red [Faithfull et al., 2012], DBtk [Bacci et al., 2009], Bigrapher [Sevegnani and Calder, 2016] et BigMC [Perrone et al., 2012]. Au cours de l'exploitation de ces outils, nous avons été confrontés à plusieurs limites : (1) outils dans un état expérimental et pas de suite dans leur développement ; (2) manipulation contraignante et manque de support de quelques concepts bigraphiques ; (3) problème lors de l'exécution des règles de réaction bigraphiques (*"matching problem"* [Birkedal et al., 2007] –c'est-à dire, identifier quelle règle de réaction doit-on appliquer pour la réécriture d'un bigraphe donné ; (4) bien que BigMC est l'unique outil pour la vérification des bigraphes, il reste très restrictif dans l'expression des propriétés désirées (aucune possibilité pour l'expression des propriétés en LTL).

Par conséquent, et à l'issue de l'étude réalisée dans le chapitre 3, nous avons constaté que l'absence d'outils et environnement fiables pour l'exécution et la vérification des bigraphes, constitue un vrai handicap. Partant de ce constat,

notre contribution dans ce chapitre consiste en la proposition d'un outil "RC-Tool4Bigraphs" pour la réécriture et la vérification des bigraphes. Dans ce qui suit, nous présentons quelques détails techniques concernant l'implémentation et l'exploitation de cet outil, à travers des exemples issus des modèles proposés. Dans la section 2, nous présentons l'ensemble des modules d'implémentation et les différentes fonctionnalités de notre outil. Dans les sections 3, nous discutons la réécriture (l'exécution) et la vérification des modèles bigraphiques (CAB et CScB) définis dans les deux chapitres précédents. Enfin, une évaluation des performances de notre outil, en matière de nombre de réécritures effectuées par unité de temps, est réalisée dans la conclusion.

6.2 Présentation de l'outil "*RCTool4Bigraphs*"

L'outil "RCTool4Bigraphs" constitue la version unifiée de nombreuses versions antérieurs, que nous avons proposé pour la manipulation et l'analyse des modèles bigraphiques spécifiés. Nous avons dans un premier temps projeté les spécifications bigraphiques définies dans [Benzadri et al., 2013a] vers des spécifications exécutables basées sur le langage Maude [Benzadri et al., 2013b]. Cette version initiale nous a permis d'éditer algébriquement et d'exécuter les modèles bigraphiques des systèmes cloud [Benzadri et al., 2014b]. Un model-checker a été également implémenté dans cette version afin de vérifier les spécifications en questions [Benzadri et al., 2015]. Dans un second temps, nous avons implémenté une autre version de l'outil avec les mêmes fonctionnalités mais en utilisant le langage JAVA pour des raisons d'efficacité et de simplicité, de plus, cette version offre une interface graphique plus conviviale [Benzadri et al., 2014a]. Malgré les avantages offerts par cette version d'outil, en ce qui concerne l'exécution et l'analyse des modèles bigraphiques proposés, nous avons réalisé qu'elle ne pourrait pas être utile pour d'autre modèles conçus pour d'autres applications. Ainsi, nous visons dans ce travail à rendre cet outil plus générique pour réécrire et vérifier n'importe quel modèle bigraphique. Le "RCTool4Bigraphs" automatise le mapping entre la théorie des bigraphes et la logique de réécriture. Il a été implémenté en utilisant un couplage judicieux entre le langage formel "Maude" [Clavel et al., 2007] et le langage de haut niveau "JAVA".

6.2.1 Architecture générale de l'outil

"RCTool4Bigraphs" consiste en un moteur de réécriture implémenté dans le langage formel "Maude", et une interface graphique de modélisation développée sous le langage de haut niveau "JAVA". La figure 6.1 présente l'architecture globale de l'outil proposé.

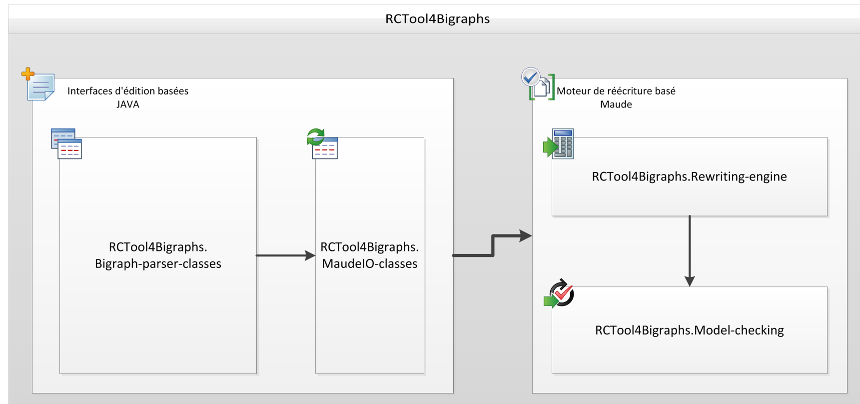


FIGURE 6.1 – Architecture du RCTool4Bigraphs

Moteur de réécriture

Le moteur de réécriture de "RCTool4Bigraphs" est implémenté sous "Maude" [Clavel et al., 2007]. Dans ce langage formel, une spécification est composée d'un ensemble de termes algébriques qui servent à l'intégration des éléments de base d'un bigraphe, et un ensemble de règles de réécriture représentant les règles de réaction bigraphiques. Nous définissons les modules systèmes suivants :

— Le module **"RCTool4Bigraphs.Rewriting-engine"**, définit la syntaxe et la sémantique des concepts de base de la théorie des bigraphes. Il se compose des trois classes ("Bigraph.maude", "PlaceGraph.maude" et "LinkGraph.maude") spécifiant l'aspect statique d'un bigraphe, et d'une classe ("BRS.maude") décrivant la dynamique des systèmes réactifs bigraphiques à travers la définition d'un ensemble de méta-règles de réaction.

— Le module **"RCTool4Bigraphs.Model-checking"** permet l'analyse des spécifications bigraphiques obtenues. Il se compose de deux modules : "BigraphPreds.maude" pour la spécification des prédicats atomiques utilisés dans description des propriétés LTL, et "BigraphChecker.maude" pour la vérification des propriétés relatives aux systèmes réactifs bigraphiques.

Table 6.1 – Bigraphe vers Maude

Théorie des bigraphes	**Logique de réécriture**	**Code source "Maude"**
Noeuds et Contrôles	Termes de sorte V et Control	sorts V. sorts Control .
ctrl	Opérateur	op _ :_ : Qid Control ->V [ctor] .
link	Opérateur	op link<_,_>=<_> : Qid V V ->LinkGraph [ctor] . op null : ->LinkGraph [ctor] .
prnt	Opérateur	op prnt<_>=<_> : V V ->PlaceGraph [ctor] . op null : ->PlaceGraph [ctor] .
Bigraphe	Terme de sorte Bigraph	sorts Bigraph PlaceGraph LinkGraph . subsorts PlaceGraphe <Bigraph . subsorts LinkGraphe <Bigraph .
Règles de réaction	Règles de réécriture	—//Exemples rl [PlaceChange] : prnt<'n1 : IaaS >=<'n2 : SaaS >=> prnt<'n1 : IaaS >=<'n3 : SaaS >. rl [LinkChange] : link<'e1 - 's1 : IaaS >=<'s2 : SaaS >,=> link<'e1 - 's1 : IaaS >=<'s3 : SaaS >.

Le tableau 6.1 présente quelques codes sources utilisés pour coder la correspondance entre la théorie des bigraphes et la logique de réécriture via son langage "Maude". Les contrôles et les nœuds sont définis comme étant des termes issus des types spécifiques (*"sorts V"* et *"sorts Control"*) dans "Maude". Aussi, la fonction de parenté (*prnt*) du graphe de places et celle de liens (*link*) du graphe de liens, sont exprimées en tant que opérateurs (*"op prnt"* et *"op link"*) dans "Maude". Enfin, les règles de réaction bigraphiques sont mises en œuvre via des règles de réécriture dans "Maude".

Interface de modélisation

L'interface de modélisation de "RCTool4Bigraphs" consiste en un ensemble de classes regroupées dans les packages JAVA suivants :

— Le package **"RCTool4Bigraphs.Bigraph-parser-classes"** est responsables de l'analyse syntaxique, l'édition et la visualisation graphique des spécifications bigraphiques entrées par l'utilisateur. Il contient les classes suivantes : "Bigraph.java", "PlaceGraph.java", "LinkGraph.java", en plus d'autres classes.

— Le package **"RCTool4Bigraphs.MaudeIO-classes"**, offre la possibilité de communiquer avec le moteur de réécriture et le modèle-checker implémentés sous "Maude". Il envoie les spécifications en questions et reçoit les résultats de réécriture et de vérification.

Table 6.2 – Bigraphe vers JAVA

Théorie des bigraphes	Langage JAVA	Code source "JAVA"
Nœuds	Classe	public class Node implements Serializable{ public String name; public Control control; public Node(String name, Control control) {....} ...}
Contrôles	Classe	public class Control implements Serializable { public String name; public String activity; public int np; public Control(String name, String activity, int np) {...} ...}
Graphe de places	Classe	public class PlaceGraph { public ArrayList liste; public Graph placegraphe; private String place; public void addN(String t, int x) {..} public void visGP() {..} ...}
Graphe de liens	Classe	public class LinkGraph { public ArrayList liste; public Graph linkgraphe; private String link; public void addL(String t, int x) {..} public void visGL() {..} ...}
Bigraphe	Classe	public class Bigraph implements Serializable { public static int ninnernames; public static int nouternames; public static int negdes; public static int nroots; public static int nsites; public PlaceGraph placegraphS; public LinkGraph linkgraphS; ....}
Règles de réaction	/	/

Le tableau 6.2 recense quelques morceaux de code source essentiel pour coder les bigraphes dans JAVA.

6.2.2 Fonctionnalités de l'outil

"RCTool4Bigraphs" offre trois fonctionnalités principales : édition, réécriture (exécution) et vérification des modèles basés bigraphes. La figure 6.2 illustre l'interface principal de notre outil.

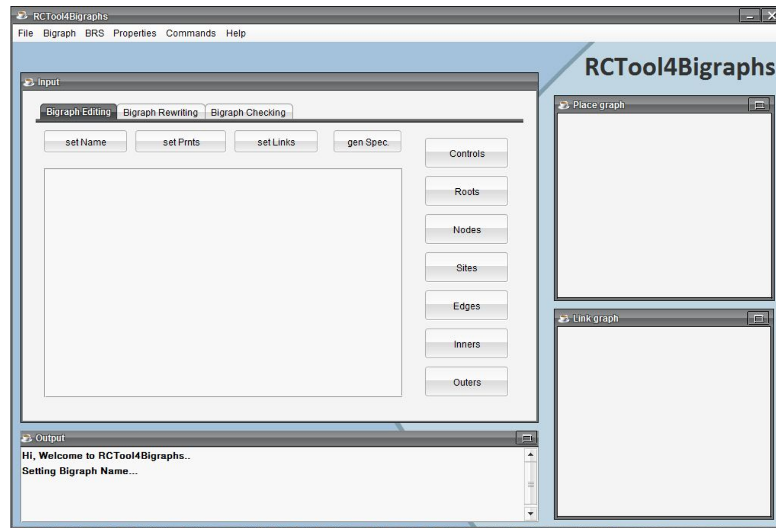


FIGURE 6.2 – Interface de modélisation principale du "RCTool4Bigraphs"

Édition des bigraphes

RCTool4Bigraphs offre une édition assistée des modèles bigraphiques, via deux modes de spécification : "Algébrique" et "Graphique". Il permet la spécification de la structure statique d'un bigraphe, tout en supportant les concepts de base de ses deux constituants : graphe de places et graphe de liens.

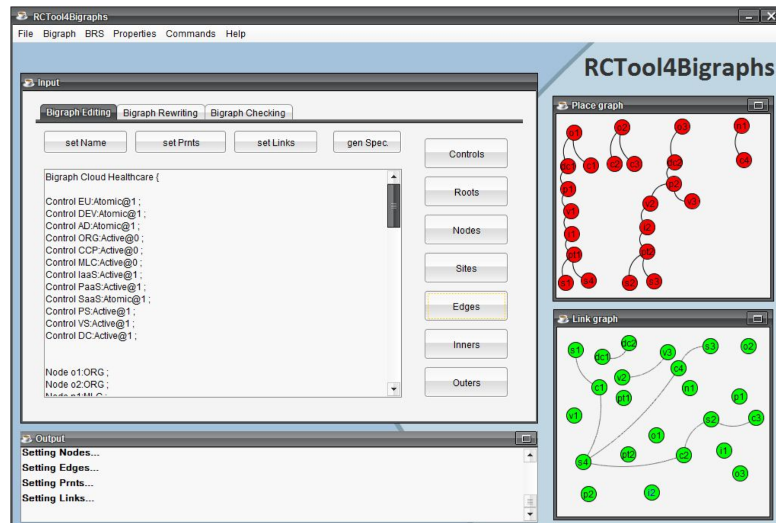


FIGURE 6.3 – Perspective "Bigraph Editing" offerte par "RCTool4Bigraphs" (pour le cas "Cloud-Healthcare")

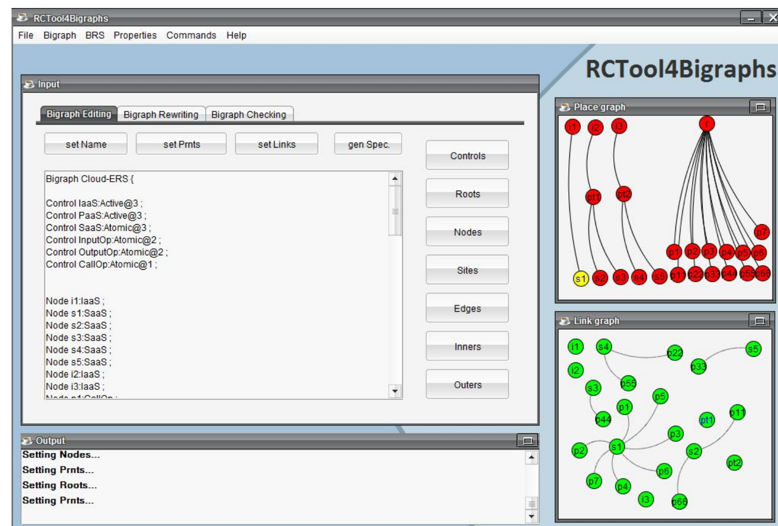


FIGURE 6.4 – Perspective "Bigraph Editing" offerte par "RCTool4Bigraphs" (pour le cas "Cloud-ERS")

Comme schématisé dans la figure 6.3 –de l'étude de cas "Cloud-Healthcare" et la figure 6.4 –de l'étude de cas "Cloud-ERS", la palette sur le côté droit de la perspective "Bigraph Editing"permet un accès rapide à l'ensemble des éléments utilisés pour la définition d'un nouveau bigraphe (Contrôles, Nœuds, etc.). Les deux vues sur la droite permettent respectivement de visualiser le graphe de places et le graphe de liens, associés aux spécifications entrées. La barre de menu en haut, contient une liste de raccourcies des différentes fonctionnalités offertes. Le menu "File" permet la création, l'ouverture et la sauvegarde d'un projet bigraphique, alors que le menu "Bigraph" offre les commandes essentielles pour la création de nouvelles instances d'un bigraphe donné.

Réécriture des bigraphes

RCTool4Bigraphs intègre un moteur de réécriture très performant ; permettant d'exécuter un ensemble de règles de réaction applicables sur le graphe de places et le graphe de liens séparément, avec la possibilité de prendre en compte les deux graphes en même temps.

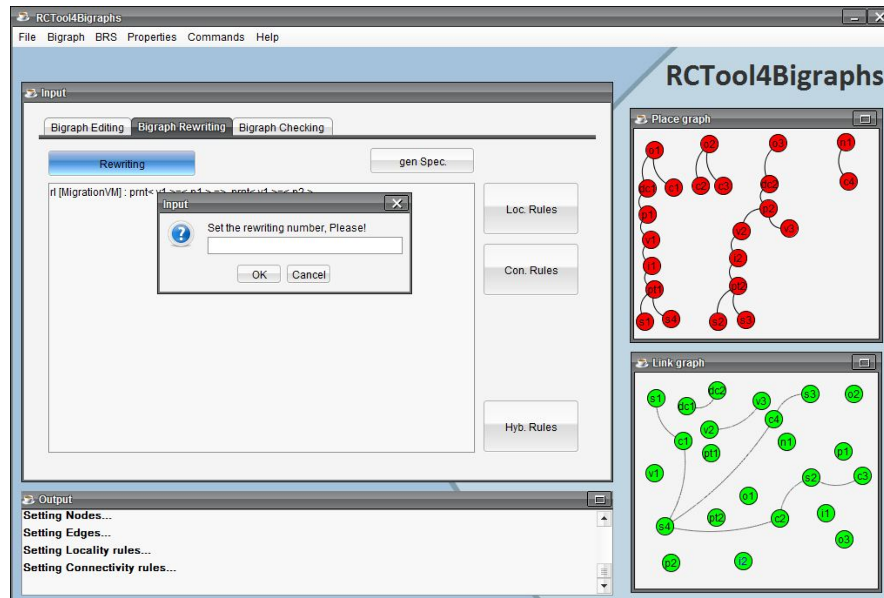


FIGURE 6.5 – Perspective "Bigraph Rewriting" offerte par "RCTool4Bigraphs"

La perspective "Bigraph Rewriting" (comme la montre la figure 6.5) permet à l'utilisateur de spécifier et d'exécuter les règles de réaction décrivant la dynamique des systèmes réactifs bigraphiques. La palette sur la droite fournit l'ensemble des éléments utilisés pour exprimer les deux catégories de règles de réaction bigraphiques ("Locality rules" et "Connectivity rules"). De plus, le menu "BRS" (de la barre de menu en haut) permet un accès rapide vers les fonctionnalités relatives à la réécriture d'un bigraphe. Le volet inférieur représente une console Maude pour afficher les divers résultats de réécriture et de vérification.

Vérification des bigraphes

RCTool4Bigraphs dispose d'un modèle checker LTL basé Maude [Clavel et al., 2007], offrant la possibilité de vérifier le comportement des systèmes spécifiés en utilisant les bigraphes. Ceci est réalisé à travers la satisfaction d'un ensemble de propriétés bigraphiques ; introduites par l'utilisateur, sous forme de prédicats.

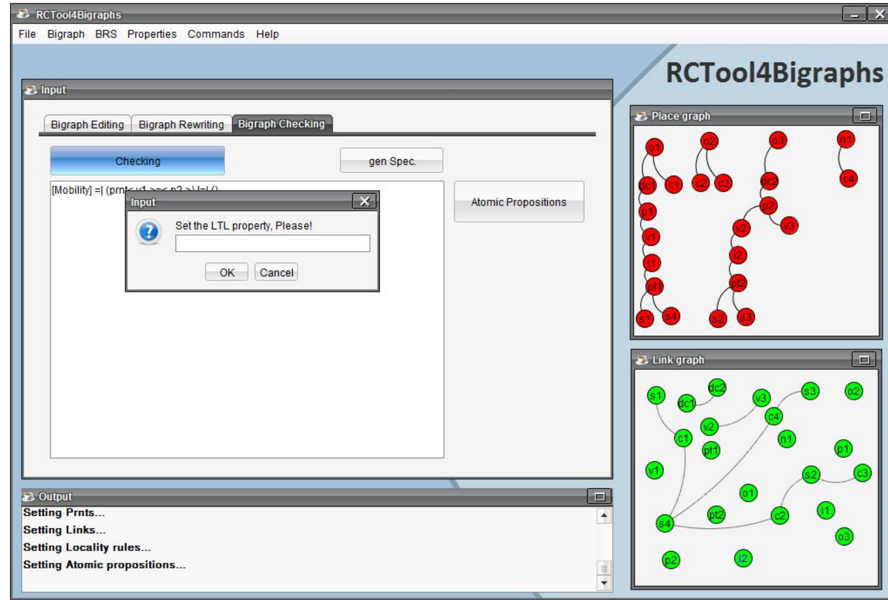


FIGURE 6.6 – Perspective "Bigraph Checking" offerte par "RCTool4Bigraphs"

La perspective "Bigraph Checking" (voir figure 6.6) présente l'interface offerte par "RCTool4Bigraphs" pour la vérification des bigraphes. La palette sur le côté droit contient les raccourcis nécessaires pour l'édition des prédicats utilisés dans la description des propriétés désirées. Ces propriétés seront vérifiées via le lancement de la commande "Checking" disponible dans le menu "Properties" de la barre de menu située en haut. Par la suite, les résultats obtenus sont affichés dans le volet inférieur.

6.3 Expérimentation de l'outil "*RCTool4Bigraphs*"

Dans cette section, nous illustrons le principe de fonctionnement de notre outil "RCTool4Bigraphs" à travers les deux études de cas : "Cloud-Healthcare" du chapitre 4 et "Cloud-ERS" du chapitre 5. A cette fin, nous proposons un processus pour la modélisation et l'analyse des systèmes cloud en générale. Ce processus vise à fournir une méthodologie éprouvée facilitant l'application des modèles théoriques définis ("CAB" et "CScB"). La figure 6.7 représente le schéma de ce processus, dont il est principalement constitué des étapes suivantes :

Phase de spécification. La phase de spécification permet la représentation de la structure statique d'un système cloud à l'aide des modèles définis : le bigraphe *"CAB"* et ses constituants *"CUB, CSB et CVB"*, ou bien le bigraphe *"CScB"* de la composition des services cloud. Cette phase offre également la possibilité de décrire l'évolution dynamique du système

spécifié, à travers les règles de réaction *"CUR, CSR et CVR"* associées au bigraphe *"CAB"*, ou bien les règles *"CVcR et CHcR"* applicables sur le bigraphe *"CScB"*.

Phase d'exécution. Conformément aux règles de réaction définies dans la première phase, la phase d'exécution permet la réécriture des spécifications bigraphiques relatives au modèle *"CAB"* ou celles du modèle *"CScB"*.

Phase de vérification. A partir des spécifications exécutables *"RCTool4Bigraphs-based specifications*"* –résultat de la deuxième phase, la phase courante permet la vérification basée "model-checking" d'un ensemble de propriétés LTL introduit par le designer.

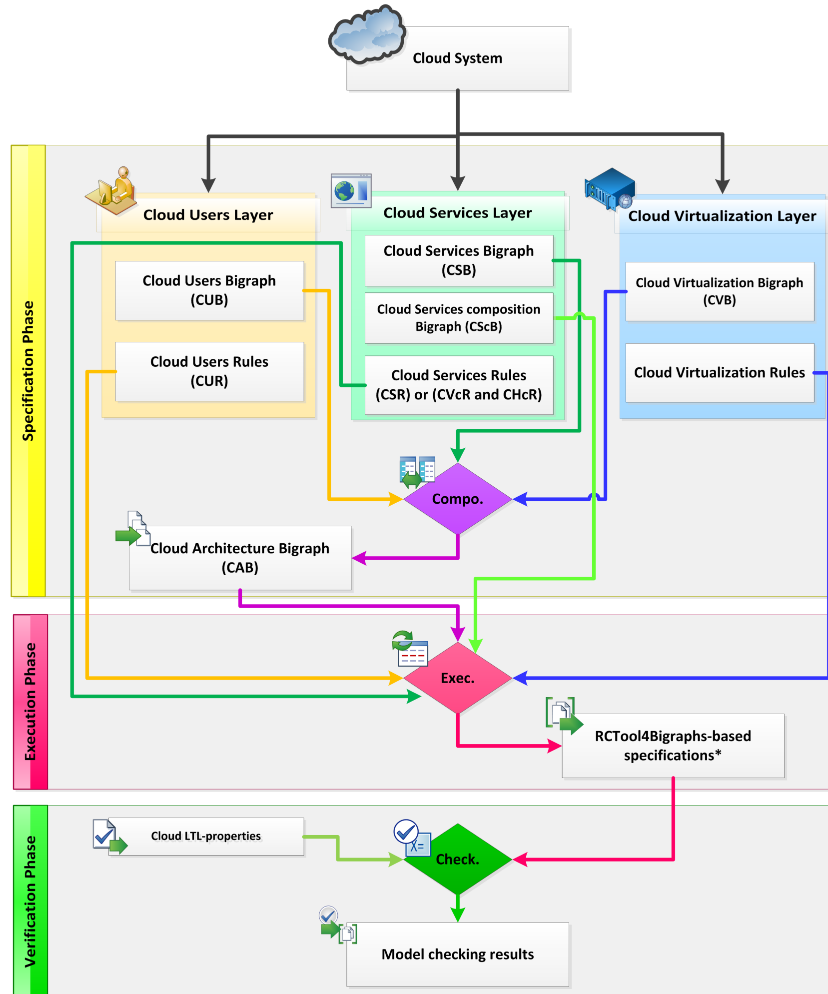


FIGURE 6.7 – Processus de modélisation et d'analyse des systèmes Cloud [Benzadri et al., 2016]

Dans ce qui suit, nous présentons d'une manière détaillée l'exécution et la vérification des modèles "CAB et CScB" en utilisant l'outil proposé "RCTool4Bigraphs", et conformément au processus suggéré.

6.3.1 Exécution et Prototypage des modèles "*CAB*" et "*CScB*"

Le noyau de "RCTool4Bigraphs" consiste en un moteur de réécriture à haute performance qui permet, à partir d'un état initial donné, de simuler le comportement d'un système spécifié en se référant aux règles de réaction définies. Pour illustrer l'exécution et le prototypage de notre outil, nous considérons les deux études de cas "Cloud-Healthcare" et "Cloud-ERS" présentées respectivement dans les chapitres 4 et 5.

Une fois les deux projets créés, les modèles bigraphiques introduits ("CAB" et "CScB") et les règles de réaction (associées à chaque modèle) spécifiées, le "RCTool4Bigraphs" procède à l'application des règles de réaction appropriées sur l'état initial spécifié, ensuite affichera l'état final obtenu après un nombre de réécriture bien déterminé. La figure 6.8 donne un aperçu des résultats d'exécution relatifs à l'étude de cas "Cloud-Healthcare" du chapitre 4. alors que la figure 6.9 illustre les résultats d'exécution relatives à l'étude de cas "Cloud-ERS" du chapitre 5.

```
RCTool4Bigraphs
File  Bigraph  BRS  Properties  Commands  Help
Output
                Copyright 1997-2010 SRI International
                       Sun Jul 24 21:00:34 2016
Maude> Maude> ==========================================
frewrite in Bigraph : prnt< dc1 >=< o1 > ; prnt< c1 >=< o1 > ; prnt< p1 >=< dc1
   > ; prnt< v1 >=< p1 > ; prnt< i1 >=< v1 > ; prnt< pt1 >=< i1 > ; prnt< s1
   >=< pt1 > ; prnt< s4 >=< pt1 > ; prnt< c2 >=< o2 > ; prnt< c3 >=< o2 > ;
   prnt< dc2 >=< o3 > ; prnt< p2 >=< dc2 > ; prnt< v2 >=< p2 > ; prnt< v3 >=<
   p2 > ; prnt< i2 >=< v2 > ; prnt< pt2 >=< i2 > ; prnt< s2 >=< pt2 > ; prnt<
   c4 >=< n1 > ; prnt< s3 >=< pt2 > ; (((((((link< e4,s4 >=< c2 > ; link< e6,
   dc1 >=< dc2 >) ; link< e5,v2 >=< v3 >) ; link< e4,s4 >=< c1 >) ; link< e4,
   s4 >=< c4 >) ; link< e3,s3 >=< c4 >) ; link< e2,s2 >=< c2 >) ; link< e2,s2
   >=< c3 >) ; link< e1,s1 >=< c1 > .
*********** rule
rl prnt< v1 >=< p1 > => prnt< v1 >=< p2 > [label MigrationVM] .
empty substitution
prnt< v1 >=< p1 >
--->
prnt< v1 >=< p2 >
rewrites: 1 in 3609213641ms cpu (0ms real) (0 rewrites/second)
result Bigraph: prnt< dc1 >=< o1 > ; prnt< dc2 >=< o3 > ; prnt< p1 >=< dc1 > ;
   prnt< p2 >=< dc2 > ; prnt< v1 >=< p2 > ; prnt< v2 >=< p2 > ; prnt< v3 >=<
   p2 > ; prnt< i1 >=< v1 > ; prnt< i2 >=< v2 > ; prnt< pt1 >=< i1 > ; prnt<
   pt2 >=< i2 > ; prnt< s1 >=< pt1 > ; prnt< s2 >=< pt2 > ; prnt< s3 >=< pt2 >
   ; prnt< s4 >=< pt1 > ; prnt< c1 >=< o1 > ; prnt< c2 >=< o2 > ; prnt< c3 >=<
   o2 > ; prnt< c4 >=< n1 > ; link< e1,s1 >=< c1 > ; link< e2,s2 >=< c2 > ;
   link< e2,s2 >=< c3 > ; link< e3,s3 >=< c4 > ; link< e4,s4 >=< c1 > ; link<
   e4,s4 >=< c2 > ; link< e4,s4 >=< c4 > ; link< e5,v2 >=< v3 > ; link< e6,dc1
   >=< dc2 >
Maude> Bye.
```

FIGURE 6.8 – Résultats d'exécution relatifs à l'étude de cas"Cloud-Healthcare"

Pour l'étude de cas "Cloud-Healthcare" (voir figure 6.8), nous reprenons l'exemple de migration des serveurs virtuels présenté dans la section 4.6. Dans un premier temps, le serveur virtuel *"v1"* est déployé à l'intérieur du serveur physique *"p1"* ($prnt < v1 >=< p1 >$) au sein du centre de données *"dc1"* ($prnt < p1 >=< dc1 >$), qui est situé dans les locaux du laboratoire d'analyses médicales *"o1"* ($prnt < dc1 >=< o1 >$). Puis, en déclenchant la commande de réécriture, le serveur virtuel *"v1"* migre vers le serveur physique *"p2"* ($prnt < v1 >=< p2 >$) au sein du centre de données *"dc2"* ($prnt < p2 >=< dc2 >$), qui est situé dans les locaux du fournisseur de services cloud *"o3"* ($prnt < dc2 >=< o3 >$).

```
RCTool4Bigraphs
File  Bigraph  BRS  Properties  Commands  Help
Output
empty substitution
link< ea4,p4 >=< p4 >
--->
link< ea4,p4 >=< p44 >
*********** rule
rl link< ea5,p5 >=< p5 > => link< ea5,p5 >=< p55 > [label CHcR5] .
empty substitution
link< ea5,p5 >=< p5 >
--->
link< ea5,p5 >=< p55 >
*********** rule
rl link< ea6,p6 >=< p6 > => link< ea6,p6 >=< p66 > [label CHcR6] .
empty substitution
link< ea6,p6 >=< p6 >
--->
link< ea6,p6 >=< p66 >
rewrites: 6 in 3609217015ms cpu (2ms real) (0 rewrites/second)
result Bigraph: prnt< s1 >=< i1 > ; prnt< s2 >=< pt1 > ; prnt< s3 >=< pt1 > ;
   prnt< s4 >=< pt2 > ; prnt< s5 >=< pt2 > ; prnt< pt1 >=< i2 > ; prnt< pt2
   >=< i3 > ; prnt< p1 >=< r > ; prnt< p2 >=< r > ; prnt< p3 >=< r > ; prnt<
   p4 >=< r > ; prnt< p5 >=< r > ; prnt< p6 >=< r > ; prnt< p7 >=< r > ; prnt<
   p11 >=< r > ; prnt< p22 >=< r > ; prnt< p33 >=< r > ; prnt< p44 >=< r > ;
   prnt< p55 >=< r > ; prnt< p66 >=< r > ; link< e1,s1 >=< p1 > ; link< e1,s1
   >=< p2 > ; link< e1,s1 >=< p3 > ; link< e11,s1 >=< p4 > ; link< e11,s1 >=<
   p5 > ; link< e111,s1 >=< p6 > ; link< e111,s1 >=< p7 > ; link< e2,s2 >=<
   p11 > ; link< e22,s2 >=< p66 > ; link< e222,s3 >=< p44 > ; link< e4,s4 >=<
   p22 > ; link< e44,s4 >=< p55 > ; link< e5,s5 >=< p33 > ; link< ea1,p1 >=<
   p11 > ; link< ea2,p2 >=< p22 > ; link< ea3,p3 >=< p33 > ; link< ea4,p4 >=<
   p44 > ; link< ea5,p5 >=< p55 > ; link< ea6,p6 >=< p66 >
Maude> Bye.
```

FIGURE 6.9 – Résultats d'exécution relatifs à l'étude de cas"Cloud-ERS"

Pour l'étude de cas "Cloud-ERS" (voir figure 6.9), nous reprenons l'exemple de composition dynamique des services cloud (application de la méta-règle de réaction CHcR) présenté dans la section 5.5. Dans un premier temps, le service composite *"s1"* fournit une seule opération *"p7"* ($link < e111, s1 >=< p7 >$), et demande six autres opérations *"p1, p2, p3, p4, p5 et p6"* ($link < e1, s1 >=< p1 >, etc.$). A noter que pour chaque opération demandée *"pi"* il existe une opération *"pii"* fournie par un service cloud disponible. Après six réécritures, les opérations demandées *"p1, p2, p3, p4, p5 et p6"* par le service composite *"s1"* sont correctement associées aux opérations fournies *"p11, p22, p33, p44, p55 et p66"* par les services cloud *"s2, s3, s4 et s5"* ($link < ea1, p1 >=< p11 >, etc.$).

6.3.2 Vérification basée Model-Checking des modèles ”*CAB*” et ”*CScB*”

La vérification formelle est une phase essentielle dans la modélisation des systèmes cloud, elle vise à déterminer si le système modélisé est bien conçu, sans erreur et de haute qualité. Un avantage majeur de l'utilisation de ”Maude” comme langage de mise en œuvre pour notre outil, est l'exploitation de son ”LTL Model-Checker” [Clavel et al., 2007]. Sur cette base, ”RCTool*4*Bigraphs” permet de vérifier que les spécifications bigraphiques introduites satisferont certaines propriétés inhérentes, ou bien d'obtenir un contre-exemple montrant que la propriété en question a été violée.

Le tableau 6.3 présente quelques propriétés de vérification (relatives aux exemples considérés) et leur formalisation via des formules LTL. Nous soulignons ici que cet ensemble de propriétés peut être enrichi par d'autres propriétés, en fonction des besoins des concepteurs.

Table 6.3 – Quelques propriétés de vérification

Propriété	**Description**	**Spécification en LTL**
”Availability”	Cette propriété est satisfaite si le modèle du système (dans ses états finaux canoniques) ne contient pas de demandes de services insatisfaites.	[] <>*not* requesting
”Mobility”	La mobilité est la capacité de déplacer les services cloud ou les serveurs virtuels d'un centre de données à un autre.	[] <>moving
”Auto-scaling”	Cette propriété est satisfaite si le nombre de serveurs peut augmenter ou diminuer dynamiquement afin de maintenir la qualité de service cloud.	[] (Increasing v decreasing)

A titre d'exemple, et après l'édition et la réécriture des spécifications bigraphiques, nous exploitons l'outil pour vérifier que nos deux modèles proposés (”CAB” et ”CScB”) satisferont deux propriétés importantes (parmi celles présentées dans le tableau 6.3).

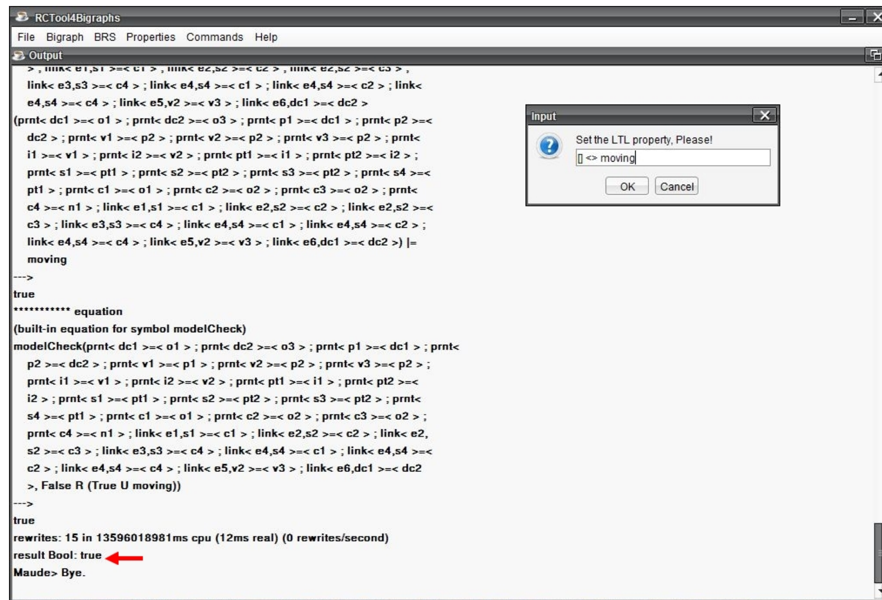


FIGURE 6.10 – Résultat de vérification de la propriété "Mobility"

Nous concluons que le modèle CAB vérifie la propriété de "Mobility" souhaitée, tel qu'il est montré sur la figure 6.10.

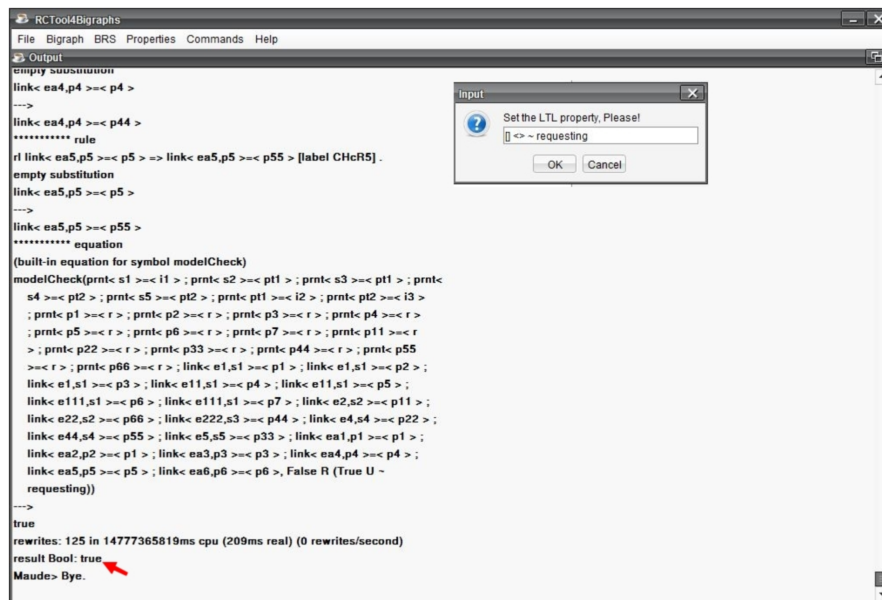


FIGURE 6.11 – Résultat de vérification de la propriété "Availability"

La figure 6.11 affiche le résultat de vérification de la propriété "Availability"

sur le modèle "CScB", qui est positif.

6.4 Conclusion

Dans ce chapitre, nous avons présenté un outil appelé "RCTool4Bigraphs" (Rewriting and Checking Tool for Bigraphs), dédié à l'édition, la réécriture et la vérification basée "model-checking" des bigraphes. "RCTool4Bigraphs" est le résultat de plusieurs tentatives d'implémentation des modèles théoriques proposés dans le cadre de cette thèse. Le but essentiel de "RCTool4Bigraphs" est de fournir un outil générique permettant la modélisation et l'analyse de n'importe quel modèle bigraphique. Enfin, en comparant avec les outils existants, "RCTool4Bigraphs" se veut le plus efficace et le très performant. La figure 6.12 met en évidence les performances de "RCTool4Bigraphs", en termes de temps et nombre de réécritures, lors de l'exécution des spécifications "CAB" relatives à l'étude de cas "Cloud-Healthcare" et celles du "CScB" relatives à l'étude de cas "Cloud-ERS". A noter que l'évaluation de notre outil a été réalisée en utilisant un PC de bureau, dont la configuration est la suivante :

— Processeur : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz 2.40GHz ;
— RAM : 4.00 GB ;
— Systeme d'exploitation : Windows 7 Ultimate (64-bit).

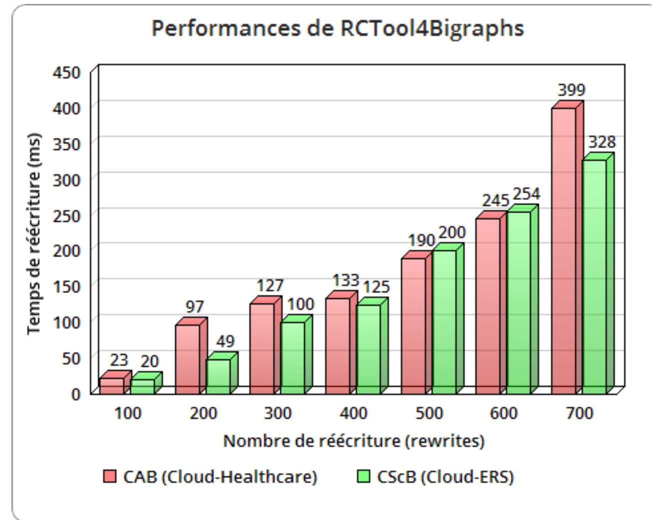


FIGURE 6.12 – Performances du "RCTool4Bigraphs"

Conclusion générale

Le Cloud Computing devient aujourd'hui le modèle informatique le plus utilisé pour héberger et exécuter les services informatiques. Ce nouveau paradigme améliore la flexibilité et l'accessibilité aux ressources cloud, et permet de réduire le coût total de possession des systèmes informatiques. Néanmoins, et en raison de leur hétérogénéité, les systèmes Cloud connaissent une croissance exponentielle en termes de taille et de complexité. Ceci engendre une définition indéterminée des éléments étant impliqués dans une architecture Cloud, et ouvre la voie vers plusieurs défis qui freinent leur croissance et leur adoption. Le travail réalisé dans le cadre de cette thèse a fait face aux trois importants défis suivants : "Bugs dans les grands systèmes distribués", "Disponibilité d'un services" et "Mise à l'échelle rapide", il a contribué à leur solution en réduisant la complexité de plus en plus contestée des systèmes Cloud.

L'objectif principal de cette thèse étant de fournir un niveau d'abstraction aux descriptions architecturales d'un système Cloud en mettant en place un modèle générique et modulaire qui permet de modéliser ses différents constituants, vu selon toutes ses facettes. Nous avons proposé une approche formelle originale, basée sur les bigraphes, pour la spécification et la vérification des systèmes Cloud. Nous avons entamé ce travail par l'état de l'art et l'étude approfondie de la littérature concernant les domaines de recherche abordés. Nous avons passé en revue les définitions universelles du Cloud Computing et l'ensemble de ses concepts importants. Ceci nous a permis de dégager une définition que nous avons adopté pour la suite de ce travail. Ensuite, nous avons comparé et classé des travaux existants de modélisation des systèmes Cloud, tout en mettant en évidence l'apport de chacun dans l'expressivité du modèle, la vérification des propriétés inhérentes, ainsi que l'expérimentation des outils dédiés. Cette étude de comparaison nous a permis, d'une part de cerner les difficultés liées à la modélisation et l'analyse des systèmes Cloud, et d'autre part de motiver l'adoption, pour la première fois dans la littérature, du formalisme des bigraphes dans la spécification et l'analyse des systèmes Cloud. Nous avons élaborée en plusieurs itérations une démarche de modélisation des systèmes Cloud tirant profit de la structure et la sémantique des systèmes réactifs bigraphiques (BRS).

Contributions

Les contributions de ce travail de thèse s'étendent de la spécification des systèmes Cloud jusqu'à leur vérification formelle et la mise en œuvre d'un outil générique pour supporter les modèles élaborés. Nous les résumons ci-dessous :

1. L'étude approfondie de l'état de l'art et les travaux existants autour de la modélisation des systèmes Cloud a grandement servi à la synthèse dégagée pour aborder correctement notre problématique.
2. Nous avons commencé par définir une architecture en trois couches (Utilisateurs, Services et Virtualisation) pour les systèmes Cloud. Cette définition modulaire permet de prendre en considération toutes les facettes d'un tel système aussi complexe qu'il soit.
3. Nous avons associé par la suite, une sémantique formelle à cette architecture à travers un modèle bigraphique (CAB "Cloud Architecure Bigraph") composé de trois bigraphes élémentaires, modélisant chacun les spécificités de la couche en question ; "Cloud Users Bigraph" supportant l'ensemble des concepts de la couche utilisateurs ; "Cloud Services Bigraph" décrivant les différents éléments de la couche services ; et "Cloud Virtualization Bigraph" concernant les entités de base de la couche virtualisation.
4. Une conséquence directe et importante qui s'est découlée de cette formalisation consiste d'une part, en la proposition d'un ensemble de méta-règles de réaction (CUR -"Cloud Users Rules", CSR -"Cloud Services Rules" et CVR -"Cloud Virtualization Rules") pour la modélisation des reconfigurations dynamiques au niveau des trois couches identifiées. Et d'autre part, la formalisation des modèles de déploiement des services Cloud.
5. Afin de procéder à la validation de notre approche, nous avons considéré une étude de cas d'un système Cloud, de taille réaliste, il s'agit de "Cloud-Healthcare".
6. Par ailleurs, nous avons porté une attention particulière à la couche Service de l'architecture Cloud pour la raffiner et exploiter au mieux son modèle bigraphique, qui a été enrichi pour donner un nouveau modèle dit : "CScB -Cloud Services composition Bigraph". Ce dernier a servi à la formalisation de la composition dynamique des services Cloud selon deux principes distincts et complémentaires : "Composition Verticale" et "Composition Horizontale". CScB a été déduit à partir d'un méta-modèle, formé d'un ensemble de classes abstraites, décrivant les différents concepts proposés autour de la composition des services Cloud.
7. Toujours et pour un souci de validation du modèle CScB, nous avons utilisé un autre système illustratif : "CloudERS" -"Cloud Emergency Response System".

8. Nous avons à la fin concrétisé tous les résultats théoriques obtenus, via l'implémentation d'un outil générique (RCTool4Bigraphs) dédié à l'édition, la réécriture et la vérification basées "Model-Checking" des bigraphes. Nous avons pour cela exécuté et analysé formellement les deux modèles proposés : CAB et CScB.

Perspectives

Les travaux que nous avons proposés ouvrent plusieurs perspectives scientifiques à court et à long terme. Nous avons l'intention d'étendre notre travail selon deux volets différents ;

D'un point de vue théorique,

— Nous projetons raffiner les deux bigraphes élémentaires modélisant respectivement la couche utilisateurs et la couche virtualisation, comme nous l'avons fait pour le bigraphe de la couche service, afin de prendre en considération d'autres aspects pertinents dans le domaine du Cloud computing.

— Nous visons étendre le processus de vérification des modèles conçus pour prendre en charge la spécification des propriétés non-fonctionnelles. Ce qui nous permettra d'effectuer une analyse quantitative, contrairement à ce qui a été réalisé dans le cadre de cette thèse (analyse qualitative), cette perspective est réalisable évidemment en enrichissant les modèles bigraphiques par des structures additives.

— Il nous semble très intéressant d'intégrer l'aspect probabiliste dans la spécification et l'analyse des systèmes réactifs bigraphiques modélisant le comportement des systèmes Cloud. Ceci peut être établi via la pondération des règles de réaction par des priorités.

— Certes, l'intégration de nos modèles bigraphiques dans Maude a été faite de manière simple et surtout générique, cependant, nous soulignons que nous avons utilisé une transformation ad-hoc, il serait plus judicieux d'établir une correspondance fondée (foncteur ou adjonction, par exemple) à base d'une étude formelle utilisant les modèles sémantiques catégoriels des deux formalismes : bigraphe et logique de réécriture.

D'un point de vue pratique,

— Nous envisageons étendre notre outil pour prendre en compte l'aspect temporel afin de caractériser l'évolution des systèmes réactifs bigraphiques. Ceci peut être réaliser à travers l'intégration de RT-Maude dans le moteur de réécriture du RCTool4Bigraphs.

Glossaire

— **CAB :** Cloud Architecture Bigraph.

— **CUB :** Cloud Users Bigraph.

— **CSB :** Cloud Services Bigraph.

— **CVB :** Cloud Virtualization Bigraph.

— **CUR :** Cloud Users reaction Rules.

— **CSR :** Cloud Services reaction Rules.

— **CVR :** Cloud Virtualization reaction Rules.

— **CScB :** Cloud Services composition Bigraph.

— **CHcR :** Cloud Horizontal composition Rule.

— **CVcR :** Cloud Vertical composition Rule.

— **RCTool4Bigraphs :** Rewriting and model-Checking Tool for Bigraphs.

Bibliographie

[Abid et al., 2013] Abid, R., Salaün, G., Bongiovanni, F., and De Palma, N. (2013). Verification of a dynamic management protocol for cloud applications. *Automated Technology for Verification and Analysis*, pages 178–192.

[Antonopoulos and Gillam, 2010] Antonopoulos, N. and Gillam, L. (2010). *Cloud computing : Principles, systems and applications*. Springer Science and Business Media.

[Armbrust et al., 2009] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., and Stoica, I. (2009). Above the clouds : A berkeley view of cloud computing.

[Bacci et al., 2009] Bacci, G., Grohmann, D., and Miculan, M. (2009). Dbtk : a toolkit for directed bigraphs. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 413–422. Springer.

[Barre and Wells, 1990] Barre, M. and Wells, C. (1990). *Category theory for computing science*, volume 49. Prentice Hall New York.

[Benlahrache, 2014] Benlahrache, N. (2014). *Définition d'un Modèle de Déploiement d'une Architecture Logicielle à Base de BRS*. Thesis.

[Benzadri et al., 2013a] Benzadri, Z., Belala, F., and Bouanaka, C. (2013a). Towards a formal model for cloud computing. In *Service-Oriented Computing–ICSOC 2013 Workshops*, pages 381–393. Springer International Publishing.

[Benzadri et al., 2013b] Benzadri, Z., Bouanaka, C., and Belala, F. (2013b). On specifying and verifying cloud systems. In *VECoS*.

[Benzadri et al., 2014a] Benzadri, Z., Bouanaka, C., and Belala, F. (2014a). Bicloud-2m : A combined bigraph maude-based tool for cloud specification and analysis. In *ICAASE*.

[Benzadri et al., 2014b] Benzadri, Z., Bouanaka, C., and Belala, F. (2014b). Verifying cloud systems using a bigraphical maude-based model checker. *Emerging Software as a Service and Analytics*, page 3.

[Benzadri et al., 2015] Benzadri, Z., Bouanaka, C., and Belala, F. (2015). A formal framework for cloud systems. *Delivery and Adoption of Cloud Computing Services in Contemporary Organizations*, page 245.

[Benzadri et al., 2016] Benzadri, Z., Bouanaka, C., and Belala, F. (2016). Bigcaf : a bigraphical-generic cloud architecture framework. *International Journal of Grid and Utility Computing*, -(-) :–.

[Bessis et al., 2012] Bessis, N., Sotiriadis, S., Xhafa, F., Pop, F., and Cristea, V. (2012). Meta-scheduling issues in interoperable hpcs, grids and clouds. *International Journal of Web and Grid Services*, 8(2) :153–172.

[Birkedal et al., 2007] Birkedal, L., Damgaard, T. C., Glenstrup, A. J., and Milner, R. (2007). Matching of bigraphs. *Electronic Notes in Theoretical Computer Science*, 175(4) :3–19.

[Birkedal et al., 2006] Birkedal, L., Debois, S., Elsborg, E., Hildebrandt, T., and Niss, H. (2006). Bigraphical models of context-aware systems. In *Foundations of software science and computation structures*, pages 187–201. Springer Berlin Heidelberg.

[Boucebsi and Belala, 2015] Boucebsi, R. and Belala, F. (2015). Towards a channels allocation scheme model for wmns based on sbrs with sharing. *MeMo 2015*, page 5.

[Bundgaard et al., 2008] Bundgaard, M., Glenstrup, A. J., Hildebrandt, T., Hojsgaard, E., and Niss, H. (2008). Formalizing higher-order mobile embedded business processes with binding bigraphs. In *Coordination Models and Languages*, pages 83–99. Springer Berlin Heidelberg.

[Buyya et al., 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6) :599–616.

[Chandramohan et al., 2012] Chandramohan, D., Vengattaraman, T., and Dhavachelvan, P. (2012). Hppc-hierarchical petri-net based privacy nominal model approach for cloud. In *2012 Annual IEEE India Conference (INDICON)*, pages 1047–1052. IEEE.

[Cherfia, 2016] Cherfia, T. A. (2016). *Un Framework Basé Bigraphes pour la Conception et l'Analyse des Systèmes Sensibles au Contexte*. Thesis.

[Cherfia et al., 2014] Cherfia, T. A., Barkaoui, K., and Belala, F. (2014). A brs-based modeling approach for context-aware systems : A case study of smart car system. In *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*, pages 310–314. IEEE.

[Clavel et al., 2007] Clavel, M., Duran, F., Eker, S., Meseguer, J., Lincoln, P., Martı-Oliet, N., and Talcott, C. (2007). All about maude. *A High-Performance Logical Framework*, 4350.

[COCIR, 2012] COCIR (2012). Advancing healthcare delivery with cloud computing. Report.

[Damgaard and Birkedal, 2005] Damgaard, T. C. and Birkedal, L. (2005). Axiomatixing binding bigraphs. Report, Citeseer.

[Damgaard and Krivine, 2008] Damgaard, T. C. and Krivine, J. (2008). A generic language for biological systems based on bigraphs. Report, Citeseer.

[Dastjerdi et al., 2010] Dastjerdi, A. V., Tabatabaei, S. G. H., and Buyya, R. (2010). An effective architecture for automated appliance management system applying ontology-based cloud discovery. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 104–112. IEEE Computer Society.

[De Win et al., 2002] De Win, B., Piessens, F., Joosen, W., and Verhanneman, T. (2002). On the importance of the separation-of-concerns principle in secure software engineering. In *Workshop on the Application of Engineering Principles to System Security Design*, pages 1–10.

[Faithfull et al., 2012] Faithfull, A., Perrone, G., and Hildebrandt, T. T. (2012). Big red : A development environment for bigraphs. In *Selected Revised Papers from the 4th International Workshop on Graph Computation Models (GCM 2012)*, volume 61.

[Fang et al., 2013] Fang, X., Wang, M., and Wu, S. (2013). A method for security evaluation in cloud computing based on petri behavioral profiles. In *Proceedings of The Eighth International Conference on Bio-Inspired Computing : Theories and Applications (BIC-TA), 2013*, pages 587–593. Springer Berlin Heidelberg.

[Fitch and Xu, 2012] Fitch, D. F. and Xu, H. (2012). A petri net model for secure and fault-tolerant cloud-based information storage. In *SEKE*, pages 333–339.

[Fortis et al., 2012] Fortis, T.-F., Munteanu, V. I., and Negru, V. (2012). Towards an ontology for cloud services. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, pages 787–792. IEEE.

[Foster et al., 2008] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. Ieee.

[Grindle et al., 2013] Grindle, M., Kavathekar, J., and Wan, D. (2013). A new era for the healthcare industry : Cloud computing changes the game. Report.

[Grohmann and Miculan, 2007] Grohmann, D. and Miculan, M. (2007). Directed bigraphs. *Electronic Notes in Theoretical Computer Science*, 173 :121–137.

[Gutierrez-Garcia and Sim, 2010] Gutierrez-Garcia, J. O. and Sim, K.-M. (2010). Self-organizing agents for service composition in cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 59–66. IEEE.

[Gutierrez-Garcia and Sim, 2013] Gutierrez-Garcia, J. O. and Sim, K. M. (2013). Agent-based cloud service composition. *Applied intelligence*, 38(3) :436–464.

[Hausman et al., 2013] Hausman, K. K., Cook, S. L., and Sampaio, T. (2013). Cloud essentials : Comptia authorized courseware for exam clo-001.

[Hojsgaard and Glenstrup, 2011] Hojsgaard, E. and Glenstrup, A. J. (2011). The bpl tool : A tool for experimenting with bigraphical reactive systems. *Bigraphical Languages and their Simulation*, page 85.

[Jangra and Bala, 2013] Jangra, A. and Bala, R. (2013). *PASA : privacy-aware security algorithm for cloud computing*, pages 487–497. Springer.

[Jardim-Goncalves et al., 2013] Jardim-Goncalves, R., Cretan, A., Coutinho, C., Dutra, M., and Ghodous, P. (2013). *Ontology enriched framework for cloud-based enterprise interoperability*, pages 1155–1166. Springer.

[Jennings, 2010] Jennings, R. (2010). *Cloud computing with the Windows Azure platform*. John Wiley and Sons.

[Jula et al., 2014] Jula, A., Sundararajan, E., and Othman, Z. (2014). Cloud computing service composition : A systematic literature review. *Expert Systems with Applications*, 41(8) :3809–3824.

[Kikuchi and Hiraishi, 2014] Kikuchi, S. and Hiraishi, K. (2014). Improving reliability in management of cloud computing infrastructure by formal methods. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–7. IEEE.

[Krivine et al., 2008] Krivine, J., Milner, R., and Troina, A. (2008). Stochastic bigraphs. *Electronic Notes in Theoretical Computer Science*, 218 :73–96.

[Kurdi et al., 2015] Kurdi, H., Al-Anazi, A., Campbell, C., and Al Faries, A. (2015). A combinatorial optimization algorithm for multiple cloud service composition. *Computers and Electrical Engineering*, 42 :107–113.

[Kurze et al., 2011] Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., and Kunze, M. (2011). Cloud federation. *CLOUD COMPUTING*, 2011 :32–38.

[Lacheheub and Maamri, 2016] Lacheheub, M. N. and Maamri, P. R. (2016). Towards a construction of an intelligent business process based on cloud services and driven by degree of similarity and qos. *Information Systems Frontiers*, pages 1–18.

[Li et al., 2011] Li, H., Liu, J., and Tang, G. (2011). A pricing algorithm for cloud computing resources. In *Network Computing and Information Security (NCIS), 2011 International Conference on*, volume 1, pages 69–73. IEEE.

[Liu et al., 2013a] Liu, D., Zhu, H., and Bayley, I. (2013a). A case study on algebraic specification of cloud computing. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 269–273. IEEE.

[Liu et al., 2013b] Liu, L., Fang, X.-w., Liu, X.-w., and Ji, J. (2013b). Study on security supervising and managing methods of the trusted cloud computing

based on petri net. In *Proceedings of The Eighth International Conference on Bio-Inspired Computing : Theories and Applications (BIC-TA), 2013*, pages 579–585. Springer Berlin Heidelberg.

[Longo et al., 2011] Longo, F., Ghosh, R., Naik, V. K., and Trivedi, K. S. (2011). A scalable availability model for infrastructure-as-a-service cloud. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks (DSN)*, pages 335–346. IEEE.

[Marks and Lozano, 2010] Marks, E. A. and Lozano, B. (2010). Executive's guide to cloud computing.

[McCarthy et al., 2006] McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4) :12.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

[Milner, 2004] Milner, R. (2004). *Bigraphs for Petri nets*, pages 686–701. Springer.

[Milner, 2005] Milner, R. (2005). Pure bigraphs. *University of Cambridge, Tech. Rep. UCAM-CL-TR-614*.

[Milner, 2009] Milner, R. (2009). The space and motion of communicating agents.

[Milner and Jensen, 2004] Milner, R. and Jensen, O. (2004). Bigraphs and mobile processes. In *Dagstuhl Seminar*, volume 4241.

[MRV-Communications, 2013] MRV-Communications (2013). A taxonomy of data communication for cloud computing.

[Muehlbauer, 2012] Muehlbauer, T. J. (2012). *Formal Specification and Analysis of Cloud Computing Management*. Thesis.

[Nassima et al., 2014] Nassima, B., Zarour, N. E., and Aknine, S. (2014). Resource management policies to increase provider's gain in a cloud coalition. *International Journal of Grid and Utility Computing*, page 130.

[Ou, 2006] Ou, G. (2006). Introduction to server virtualization. *Techrepublic. com*, 5.

[Perrone et al., 2012] Perrone, G., Debois, S., and Hildebrandt, T. T. (2012). A model checker for bigraphs. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1320–1325. ACM.

[Rezaee et al., 2014] Rezaee, A., Rahmani, A. M., Movaghar, A., and Teshnehlab, M. (2014). Formal process algebraic modeling, verification, and analysis of an abstract fuzzy inference cloud service. *The Journal of Supercomputing*, 67(2) :345–383.

[Sevegnani, 2012a] Sevegnani, M. (2012a). *Bigraphs with sharing and applications in wireless networks*. Thesis.

[Sevegnani, 2012b] Sevegnani, M. (2012b). Bigraphs with sharing and applications in wireless networks.

[Sevegnani and Calder, 2015] Sevegnani, M. and Calder, M. (2015). Bigraphs with sharing. *Theoretical Computer Science*, 577 :43–73.

[Sevegnani and Calder, 2016] Sevegnani, M. and Calder, M. (2016). Bigrapher : rewriting and analysis engine for bigraphs.

[SOAWorkGroup, 2013] SOAWorkGroup, O. (2013). Service oriented architecture : What is soa ?

[Standard, 2009] Standard, I. . (2009). Common criteria for information technology security evaluation. *The Common Criteria*.

[Van Der Waerden, 2013] Van Der Waerden, B. L. (2013). *A history of algebra : from al-Khwarizmi to Emmy Noether*. Springer Science and Business Media.

[Wang et al., 2012] Wang, F., Zhu, J., Shen, S., and Chen, M. (2012). An ontology cloud-shadow model based knowledge service framework. In *Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering*, pages 557–562. Springer.

[World-Health-Organization, 2015] World-Health-Organization (2015). World health statistics 2015.

[Wu, 2011] Wu, M. (2011). *Cloud Computing : Hype or Vision*, volume 227 of *Communications in Computer and Information Science*, book section 45, pages 346–353. Springer Berlin Heidelberg.

[YANG and WANG, 2012] YANG, S. and WANG, S. (2012). The formal description based on p-calculus for verifying the atn in cloud computing. *Journal of Computational Information Systems*, 14(8).

[Zou et al., 2010] Zou, J., Wang, Y., and Lin, K.-J. (2010). A formal service contract model for accountable saas and cloud services. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 73–80. IEEE.